

Oracle快手DBA

零基础入门实战

史跃东　编著

清华大学出版社

Oracle 快手 DBA 零基础入门实战

史跃东　编著

清华大学出版社
北　　京

内 容 简 介

本书旨在为初学者提供一本入门级书籍。使得读者可按本书中的内容，从零开始，独立完成数据库的基本安装配置、SQL 书写、数据库管理、备份恢复，并了解初步的性能优化的相关知识。本书摒弃了以往相关书籍以理论为主的写作理念，重在引导读者实际动手完成操作。

图书在版编目(CIP)数据

Oracle 快手 DBA 零基础入门实战/史跃东 编著. —北京：清华大学出版社，2016
ISBN 978-7-302-44540-1

Ⅰ. ①O… Ⅱ. ①史… Ⅲ. ①关系数据库系统 Ⅳ. ①TP311.138

中国版本图书馆 CIP 数据核字(2016)第 174421 号

责任编辑：王　军　韩宏志
封面设计：牛艳敏
版式设计：孔祥峰
责任校对：曹　阳
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62794504
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**15　　**字　　数：**346 千字
版　　次：2016 年 8 月第 1 版　　**印　　次：**2016 年 8 月第 1 次印刷
印　　数：1～3000
定　　价：39.80 元

产品编号：



前　言

2014 年底，笔者开始在天津对在校大学生进行 Oracle 技术培训。当时就有不少学生来找笔者，让笔者推荐一本较好的入门级书籍。笔者虽然长期研究 Oracle 技术，但真的去想一下，发现还没有什么比较适合初学者的 Oracle 书籍。反倒是基于 Oracle 知识领域的某一部分进行深入研究的书比较多，例如专门写备份恢复或者性能优化方面的。当然，对于有经验的 DBA 而言，翻阅这些专门关注某个方向的数据库书籍，是个很好的深入学习的方法。但是对于初学者而言，可就不太适合了。

2015 年在北京做 Oracle 认证培训的时候，又有学生来找笔者，说市面上的很多 Oracle 书籍都是基于 Windows 的，想去找一本基于 Linux 的书也不太容易。笔者以前倒是没有注意这个问题，毕竟当年笔者进入 Oracle 的大门，是通过阅读大量官方文档来实现的，几乎没怎么关注市面上 Oracle 相关的入门级书籍，也就没有注意到操作系统版本的问题。而实际上，在生产系

统中，Linux 或者类 UNIX 的操作系统才是更常见的。因此，基于这样的操作系统来学习 Oracle 知识显然更贴近实战一些。

再者，市面上很多数据库相关书籍，都侧重于理论方面。笔者并非认为注重理论就不对，但是 DBA 确实是一个极关注动手能力的职业。无论你是否精通理论，只要能把问题搞定，你就是一名优秀的 DBA。另外，对于初学者而言，一上来就面对大量枯燥的理论，也很容易对 Oracle 技术产生厌烦心理。而大家都知道，与其他数据库相比，Oracle 的入门应该是最有难度的。

于是，笔者慢慢地就有了一个想法。从零开始学习 Oracle，是否可以从动手开始，由实验反推理论，通过实验来获取结论？先让初学者自己大量动手，快速上手，在基本掌握 Oracle 的常规操作后，再深入研究理论，并与实验并重。这样，对于初学者而言，或许会更容易接受一些。

再加上前段时间经好友推荐，结识了清华大学出版社的一位编辑，在经过热烈讨论后，专门针对初学者的这本书就正式付梓出版了。

读者对象

毋庸置疑，这是一本专门面向初学者甚至是零基础人员的入门级 Oracle 书籍。

本书没有大量的枯燥理论，只有一个个经典的 Oracle 数据库实战实验。各位读者可按本书的内容，从零开始，一点一点地去完成操作系统安装、数据库软件安装及数据库创建，然后将命令一条一条地输入你的电脑。无论是 IT 从业人员，还是在校学生，甚至是没有什么计算机基础的“外行”，你都可以从这本书开始，一步一步地进入 Oracle 数据库技术的大门。

本书提倡手把手辅导，实验步骤及命令十分详明，读者可遵循这些步骤完成本书的全部实验。但是切记，笔者更希望每位读者能亲手输入本书中的命令。DBA是一个对动手能力要求极高的岗位，换言之，你的功夫都在手上。想象一下，当数据库出现故障时，在领导及同事面前，你淡定自若，手指如飞，有条不紊地将各种疑难杂症一一搞定，那该是怎样的场景？

想成为这样的高手吗？那就从阅读这本书开始吧。

勘误

虽然笔者对本书的内容进行了再三审查，但是书中依然可能存在错字错句甚至错误命令。因此，如果各位读者在阅读本书时遇到这样的问题，请随时与笔者联系。笔者的邮箱为 shiyuedong@hotmail.com，当然，也可加笔者的微信 caunique。



致　谢

首先要感谢好友王楠和清华大学出版社的王军编辑，没有你们二位，此书断然无法出版，笔者的想法也无法付诸笔端。因此，在此恭祝二位及家人身体健康，事事如意，并希望在以后能够继续合作。

当然，还要谢谢我的妻子。正是因为她，笔者才可以心无旁骛，专心完成本书的写作。

其他不少朋友或同行对本书出版亦有贡献，在此一并谢过。

史跃东



目　　录



第 1 章

概述与环境准备

20 世纪 70 年代关系模型理论的提出，距今已经 40 多年了。在此期间，关系型数据库从无到有、从小到大、从弱到强地发展起来。时至今日，市面上已经有 Oracle 公司的 Oracle 产品、微软的 SQL Server、IBM 的 DB2 等多种关系型数据库产品，此外还有开源的 MySQL、与大数据相关的 HBase、常用于缓存数据的 Redis、NoSQL 数据库 MongoDB 等。目前数据库领域已经是百家争鸣的格局了。尤其是近十年随着大数据的兴起和非规范化数据(如视频、图片等)的急剧增加，新兴数据库不断出现。这其中，国产数据库(例如武汉达梦、人大金仓、湖南上容

以及山东浪潮的 K-DB 等)也开始逐步进入相对成熟阶段，整个数据库市场更是新产品不断涌现，各种新特性争相亮相。然而拨云见日之后，Oracle 数据库依然是关系型数据库产品中最强大与突出的。

关于 Oracle 数据库的悠久历史，笔者在这里不再赘述。只想讨论一下与另外众多数据库相比，它的优势和特色：

- 功能和性能足够强大
- 开放性与可研究性

以上两个特点，应该是 Oracle 数据库最与众不同的地方了。首先，功能强大，性能优异，无论与其他任何关系型数据库产品相比，Oracle 数据库都在功能与性能上高出一筹。其次，Oracle 数据库足够开放，当然不是指开源数据库可以让用户自己修改源代码的这种开放性，而指在网络上，已有足够多的技术社区和相关论坛，以及众多的博客来提供相关的各种资料，从而使得任何人都可以极便捷地得到 Oracle 的相关信息与技术细节，并进行研究。再者，国内不少城市，例如北京、杭州、上海等地，已经有相当多的线下技术分享与沙龙活动。技术爱好者们完全可以就近参加这些活动，互相学习共同提高。

Oracle 数据库发展到现在，已然经历了多个版本变化。目前市面上，在企事业单位的实际应用中，以 Oracle 11*g* 为主流版本，尤以 11.2.0.4 为甚。但在笔者最近接触到的项目中，采用 12.1 版本的应用也日渐增多。但也有不少系统仍旧采用 10*g* 版本。因为 2013 年底，Oracle 公司就已经宣布不再对 10*g* 版本的数据库提供官方技术支持。因此对于仍旧采用老版本数据库的企业来说，还是有一定风险的。本书以 11.2.0.4 为主要环境，来演示众多实际生产系统中的经典实验案例，并在最后使用 12.1.0.2 版本来研究 12*c* 中的一些重要新特性。

1.1 数据库、数据仓库与大数据

众所周知，现在是大数据时代。这个自诞生到现在刚好 10 年的概念，至今已然席卷全球，成为各行各业讨论的热门话题，对当今人类社会的诸多方面都产生了重要影响。就在几年前，笔者的一个师姐(一位哲学博士后)，有段时间还曾跟笔者聊到大数据等话题。大数据的影响之深，波及行业之广，由此可知。

那么，可曾有人想过，大数据究竟是怎样流行起来的呢？它的前世今生又是如何？笔者并非大数据方面的专家，但恰好有幸亲历了从数据库到数据仓库，再到大数据的行业技术发展过程，故在此略加阐述。

我们知道，数据库用来存储大量数据，并处理高并发的用户访问，同时保证数据的完整性和可恢复性。数据库是企业的数据量达到一定程度后，所需的一种管理数据的技术。但再进一步呢？

一家公司可能有多个业务系统，有用于管理人力资源的，有用于支持销售的，有用于控制工厂生产的，等等。一般情况下，这些系统都有自己私有的数据库。这些数据库之间，虽然会

有数据交互和访问，但往往是零星的。那么，请设想以下需求：当公司发展到一定程度，公司总经理忽然想到，能否将这些类似于信息孤岛的诸多业务系统打通，使得可以在整个公司的层面上来查看数据？例如，要是想知道过去数年间公司销售业绩的增长曲线图，并分析它与公司产品策略、生产排产计划等之间有何种关系，那该如何处理？

这便是产生数据仓库的社会需求。数据仓库的概念在1990年由 Inmon W.H 在他的著作 *Building the Data Warehouse* 中提出。其完整概念如下：

数据仓库(Data Warehouse)是一个面向主题的(Subject Oriented)、集成的(Integrated)、相对稳定的(Non-Volatile)、反映历史变化(Time Variant)的数据集合，用于支持管理决策(Decision Making Support)。

数据仓库的出现，使得公司高层有了能够从全局把握公司全部数据的工具，从而可以在更高层次上洞察影响公司业务发展的诸多因素之间的关系，并因此做出能够影响公司战略发展的重要决策。于是自 20 世纪末至 21 世纪的前 10 年，算得上是数据仓库发展的黄金时期了。笔者也曾经在刚刚毕业的时候，参与了天津电力数据中心的建设项目。

当时的数据仓库技术，依然是基于传统的关系型数据库构建而成。底层的数据存储仍是诸如 Oracle 的数据库。但是为了处理比原有数据库中更大量的数据(数据库中的数据量，我们称为大量；数据仓库中的数据量，我们称为海量)，分区技术、并行技术等逐渐涌现，大型甚至超大型数据库以及数据中心也应运而生。那时，不少公司和政府机构都在忙于建立大型数据中心，并对其管理的海量数据进行分析和挖掘，期望能从中获得有价值的信息。

但由于当时条件所限，很多企业的历史数据积累不够，数据质量也堪忧，数据挖掘与分析的相关算法尚不完善，种种不利因素，使得当时虽然有不少企业投身于数据仓库研究与搭建，但最终能得到充分利用的，其实不多。

再往后，大数据出现了。

从数据中挖掘有价值的信息，这一由数据仓库提出的理念，被大数据完整继承下来，并进一步发扬光大。大数据采用了全新的架构和搭建技术，一开始就是为处理海量数据而生。它不但能处理传统的规范化数据，就连非规范化数据也照单全收。这顺应了近年来电商和社交类网站的兴起这一趋势，因此大数据就这样在极短时间内流行起来。而作为大数据前身的数据仓库，便就这样被大数据无情取代。

虽然现在还有一些公司在采用数据仓库技术，但是现在，确实已经是大数据的天下了。

1.2　数据库技术在大数据中的地位与价值

那么问题来了，大数据如此流行，那数据库在这样的时代，其地位又将如何？

其实很简单，大数据时代，数据库技术依然是核心技术之一。无论多大的数据量，只要涉及存储及访问处理，其原理和实现方式都是极接近的。并且 Oracle 公司本身，也在拥抱大数据

技术，他们已经提供了 Oracle 数据库和大数据/Hadoop 的应用接口。可以在 Oracle 中存储数据，然后在 Hadoop 中进行分析处理。

因此，即使是在大数据称雄天下的时代，数据库技术依然是不可或缺的。

1.3 相关技术

数据库是基于操作系统和存储之上的应用软件。因此，在实际使用和管理 Oracle 数据库系统时，也往往会涉及部分操作系统和存储的相关技术。当然，若是从整个应用系统的角度看，与数据库相关的技术就更多，如网络、中间件等。

本书旨在为数据库技术初学者或零基础人员提供入门级别的技术指导与动手实验，因此不涉及网络和中间件这样的技术。但是数据库基本的安装配置与管理，会涉及部分操作系统的相关知识，因此如果读者有一定的 Linux 基础，则学习效果更好。毕竟，本书的实验都是基于 Red Hat 6.4 搭建的环境来完成的。

当然，在本书后续内容中涉及 Linux 的地方，笔者也会详细介绍相关的知识。

1.4 本书内容与架构说明

本书的内容以 Oracle 数据库中典型的实验为主，涵盖 SQL、数据库配置管理、备份恢复、性能优化等数据库知识领域。每部分都有具体的实验内容及相关知识说明。读者可以根据这些实验一步步进入 Oracle 数据库的知识殿堂。

因此建议读者能够按照本书的章节内容，认真完成书中的每个实验。这些实验是笔者和诸多同行在多年工作经验中的实战总结，具有较高的实际应用价值。另外，在阐述如何去完成这些实验的同时，也会解释它们能够解决实际中的何种问题，适用于什么样的业务场景，以及相关的知识点等等。

1.5 实验环境准备

至此，Oracle 的相关知识已经介绍完毕，可以开始准备后续实验的环境了。可扫描本书封底的二维码，访问本书的支持网站，通过相关内容或链接，获得学习本书所需的安装介质说明和相关技术资料：

- Oracle 11gR2 安装介质说明
- Oracle 11gR2 官方文档
- Oracle 11gR2 安装手册
- Red Hat 6.4 64 位安装介质说明

请下载或按照说明获取上述所有资源，这些内容在本书的后续实验中都将用到。在下章的实验之前，我们首先需要按照文档“Oracle 11gR2 安装手册”依次完成虚拟机创建、Red Hat 6.4 操作系统安装与配置，以及数据库软件安装和监听创建工作。该安装文档的内容足够详细，请按步骤认真操作即可。



第 2 章

手工建库实验

这一章开始本书的第一个正式实验——用纯手工方式，从头开始建立一个数据库。实际上，Oracle 提供了多种创建数据库的方式，例如可以使用 dbca(该工具的使用会在后面的 5.3 节中说明)这样的图形化工具等。但使用纯手工的方式建库，可以让我们更清楚地了解数据库的详细创建过程，了解其实际要完成的任务及步骤。另外，有些管理严格的生产系统，是不允许使用图形化工具的，这时，手工方式就体现出它的价值了。

2.1 实验步骤

1) 查看监听状态

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ lsnrctl status
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 22-APR-2016 09:55:32
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date                22-APR-2016 09:49:27
Uptime                    0 days 0 hr. 6 min. 5 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/oracle/product/11.2.0/network/admin/listener.ora
Listener Log File
/u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
The listener supports no services
The command completed successfully
```

本步骤在前面 1.5 节的基础上继续进行。在前面完成监听的创建后，查看该监听的状态。注意，关于监听的功能用途及管理，我们在后面的实验中会详细说明。

2) 目录创建

```
[oracle@rhel6 ~]$ cd $ORACLE_BASE
[oracle@rhel6 oracle]$ pwd
/u01/oracle
[oracle@rhel6 oracle]$ mkdir -p admin/ora11g/adump
[oracle@rhel6 oracle]$ mkdir -p oradata/ora11g
[oracle@rhel6 oracle]$ mkdir fra
[oracle@rhel6 oracle]$ cd
[oracle@rhel6 ~]$ pwd
```

```
/home/oracle
[oracle@rhel6 ~]$ mkdir -p backup/control
```

$ORACLE_BASE 是前面 1.5 节安装的时候设置的环境变量，为 Oracle 数据库的基目录。我们安装的数据库软件及产品目录，都存放在这个基目录的子目录中。这里，我们分别在/u01/oracle 目录下创建 adump 目录，用于存放审计生成的文件，同时创建 oradata 和 fra 目录。然后在/home/oracle 目录下创建 control 目录，用于存放控制文件。关于控制文件的内容及用途，后面会详细描述。

3) 创建密码文件

```
[oracle@rhel6 ~]$ cd $ORACLE_HOME/dbs
[oracle@rhel6 dbs]$ ls
init.ora
[oracle@rhel6 dbs]$ orapwd file=orapwora11g password=oracle
[oracle@rhel6 dbs]$ ls
init.ora  orapwora11g
```

密码文件是用来管理数据库用户的。当然，实际上该文件中只存储那些拥有 sysdba 权限的用户及其密码。比如我们后面要用到的 sys 用户。该用户能够启动和关闭数据库。因此，不能把这样的用户及其密码存放在数据库中，只能存放在操作系统中。

4) 创建参数文件 pfile

与上一步在同样的目录下。执行如下操作：

```
[oracle@rhel6 dbs]$ cat init.ora |grep -v ^#|grep -v ^$ > initora11g.ora
[oracle@rhel6 dbs]$ ls
init.ora  initora11g.ora  orapwora11g
```

这里，我们利用 Oracle 提供的样例参数文件来创建新的数据库初始化参数文件。grep 命令用于过滤该样例参数文件中的注释行(以#开头的行)和空行。执行完成后，可以看到当前目录下生成了一个新文件：initora11g.ora。接下来，我们开始编辑这个文件。使用 vi 命令：

```
[oracle@rhel6 dbs]$ vi initora11g.ora
```

打开该文件，如下：

```
db_name='ORCL'
memory_target=1G
processes = 150
audit_file_dest='<ORACLE_BASE>/admin/orcl/adump'
audit_trail ='db'
db_block_size=8192
db_domain=''
```

```
db_recovery_file_dest='<ORACLE_BASE>/flash_recovery_area'
db_recovery_file_dest_size=2G
diagnostic_dest='<ORACLE_BASE>'
dispatchers='(PROTOCOL=TCP) (SERVICE=ORCLXDB)'
open_cursors=300
remote_login_passwordfile='EXCLUSIVE'
undo_tablespace='UNDOTBS1'
control_files = (ora_control1, ora_control2)
compatible ='11.2.0'
```

我们需要修改该文件的内容，输入 i，进入插入模式(当前窗口最下方出现--INSERT——字样)。该参数文件修改后的内容如下：

```
db_name='ora11g'
memory_target=800M
processes = 150
audit_file_dest='/u01/oracle/admin/ora11g/adump'
audit_trail ='db'
db_block_size=8192
db_domain=''
db_recovery_file_dest='/u01/oracle/fra'
db_recovery_file_dest_size=4G
diagnostic_dest='/u01/oracle'
open_cursors=300
remote_login_passwordfile='EXCLUSIVE'
undo_tablespace='UNDOTBS1'
control_files =
(/u01/oracle/oradata/ora11g/control01.ctl,/u01/oracle/fra/control2.ctl)
compatible ='11.2.0'
```

其中，阴影着重显示部分为修改的内容。

修改完毕后，保存退出。

注意，这里每行中需要修改的内容都需要仔细确认，否则后面的步骤可能报错。

5) 创建参数文件 spfile

```
[oracle@rhel6 dbs]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Fri Apr 22 10:39:57 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> create spfile from pfile;
File created.
SQL> startup nomount;
```

```
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size                2257840 bytes
Variable Size             490736720 bytes
Database Buffers          339738624 bytes
Redo Buffers              2371584 bytes
```

本步骤完成 spfile 的创建，并启动数据库实例。关于 pfile/spfile 以及何为数据库实例，本章最后将进行说明。

6) 执行创建数据库的命令

新打开一个窗口：

```
[root@rhel6 ~]# su - oracle
[oracle@rhel6 ~]$ cd
[oracle@rhel6 ~]$ vi create_db.sql;
```

这里，我们需要用到前面1.5节中下载的官方文档(E11882_01.rar)。下载该压缩包后，进行解压，会生成一个同名的目录。然后进入该目录，找到 index.htm 文件，双击打开，如图2-1所示：

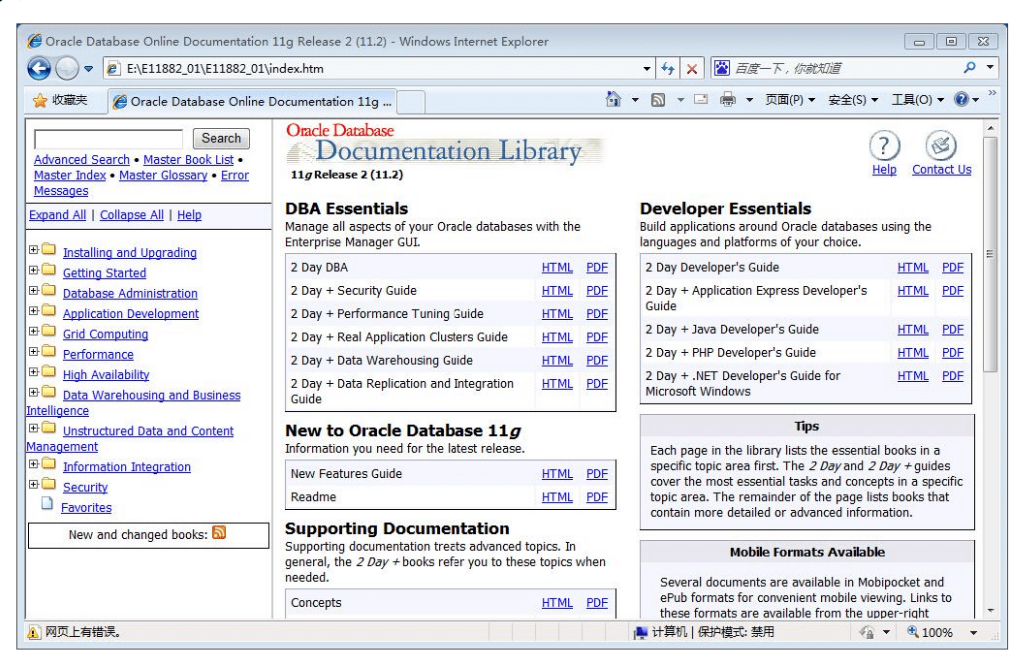


图 2-1　官方文档起始页

可将该地址保存到浏览器的收藏夹中，这样以后就可以随时使用了。

点击该页面左上角的 Master Book List 标签，进入如下页面(参见图 2-2)：

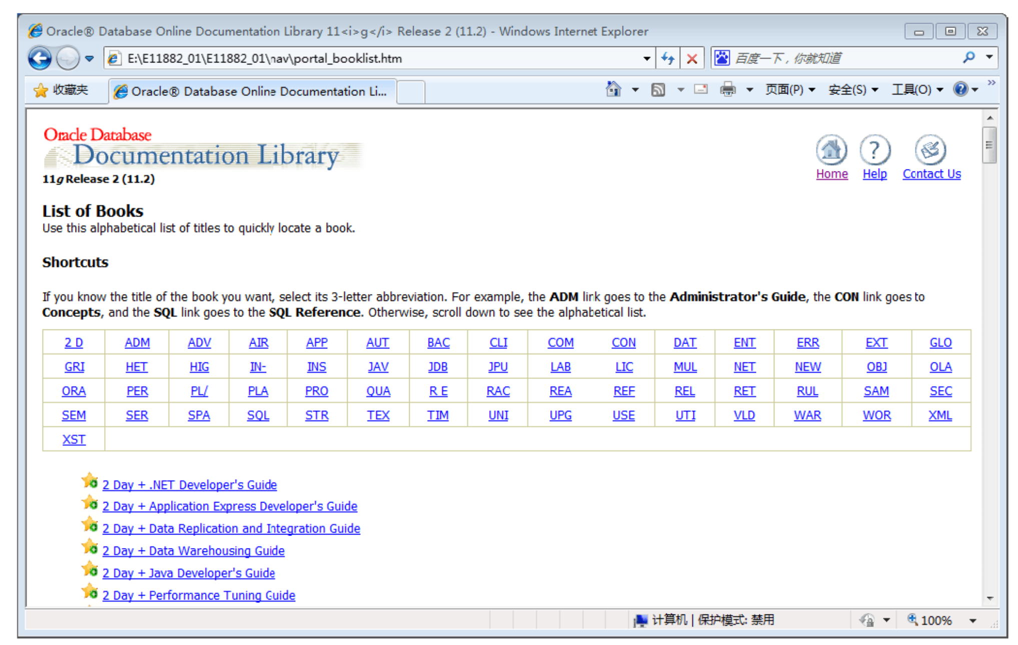


图 2-2　文档列表

这里，我们需要查看的官方文档是 *Oracle Database Administrator's Guide*，因此点击本页面中的 ADM 标签，进入如下页面(参见图 2-3)：

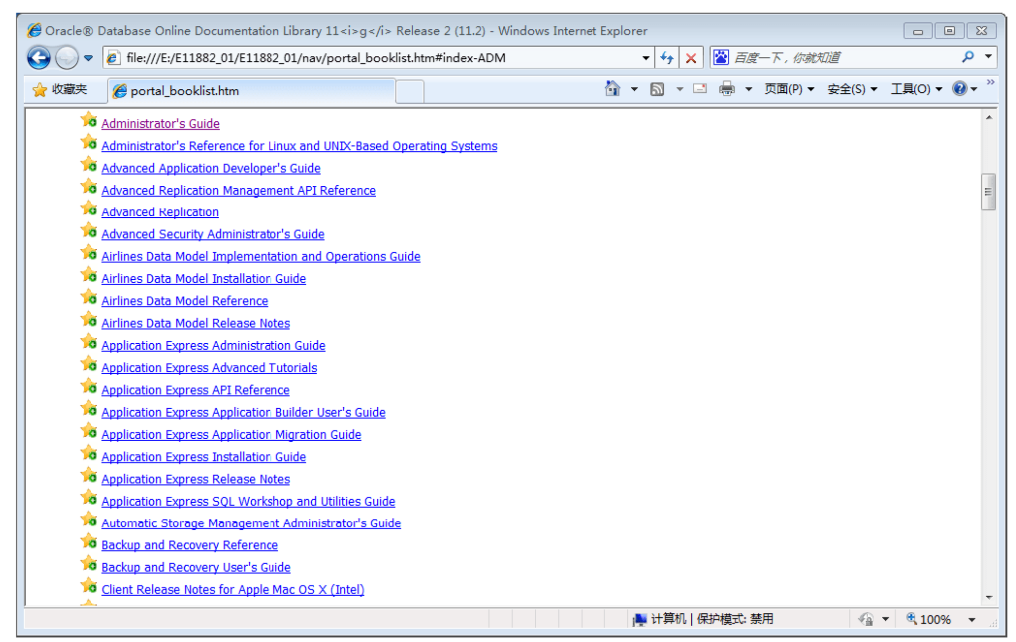


图 2-3　管理员手册页面

该页面中的第一个文档就是我们想要的。点击进入(参见图 2-4)：

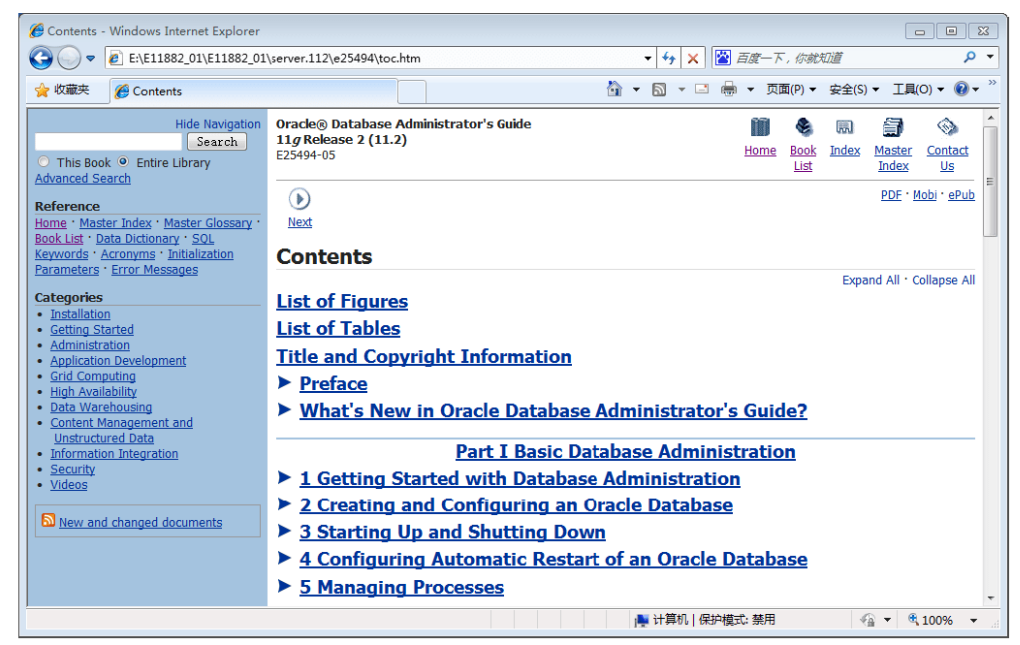


图 2-4　管理员手册内容页

这份文档是 Oracle 数据库的管理员手册，里面记录了 Oracle 数据库的诸多基本概念和相关操作命令，是我们在实际工作中经常查阅的官方文档之一。我们现在要用的是里面的第 2 章：Creating and Configuring an Oracle Database。

点击进入，再点击 Creating a Database with the CREATE DATABASE Statement，然后点击下面的 Step 9: Issue the CREATE DATABASE Statement，这里的 Example 1 下面的 CREARE DATABASE 就是我们要找的手工创建数据库的命令。如下：

```
    CREATE DATABASE mynewdb
       USER SYS IDENTIFIED BY sys_password
       USER SYSTEM IDENTIFIED BY system_password
       LOGFILE GROUP 1 ('/u01/logs/my/redo01a.log','/u02/logs/my/redo01b.log')
SIZE 100M BLOCKSIZE 512,
               GROUP 2 ('/u01/logs/my/redo02a.log','/u02/logs/my/redo02b.log')
SIZE 100M BLOCKSIZE 512,
               GROUP 3 ('/u01/logs/my/redo03a.log','/u02/logs/my/redo03b.log')
SIZE 100M BLOCKSIZE 512
       MAXLOGFILES 5
       MAXLOGMEMBERS 5
       MAXLOGHISTORY 1
       MAXDATAFILES 100
       CHARACTER SET AL32UTF8
       NATIONAL CHARACTER SET AL16UTF16
```

```
    EXTENT MANAGEMENT LOCAL
    DATAFILE '/u01/app/oracle/oradata/mynewdb/system01.dbf' SIZE 325M REUSE
    SYSAUX DATAFILE '/u01/app/oracle/oradata/mynewdb/sysaux01.dbf' SIZE 325M
REUSE
    DEFAULT TABLESPACE users
      DATAFILE '/u01/app/oracle/oradata/mynewdb/users01.dbf'
      SIZE 500M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED
    DEFAULT TEMPORARY TABLESPACE tempts1
      TEMPFILE '/u01/app/oracle/oradata/mynewdb/temp01.dbf'
      SIZE 20M REUSE
    UNDO TABLESPACE undotbs
      DATAFILE '/u01/app/oracle/oradata/mynewdb/undotbs01.dbf'
      SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

将上述代码粘贴到本步骤开始创建的 create_db.sql 文件中并进行修改，修改完成后内容如下：

```
  CREATE DATABASE ora11g
    USER SYS IDENTIFIED BY oracle
    USER SYSTEM IDENTIFIED BY oracle
    LOGFILE GROUP 1
('/u01/oracle/oradata/ora11g/redo01a.log','/u01/oracle/oradata/ora11g/redo01b.
log') SIZE 100M,
           GROUP 2
('/u01/oracle/oradata/ora11g/redo02a.log','/u01/oracle/oradata/ora11g/redo02b.
log') SIZE 100M,
           GROUP 3
('/u01/oracle/oradata/ora11g/redo03a.log','/u01/oracle/oradata/ora11g/redo03b.
log') SIZE 100M
    MAXLOGFILES 50
    MAXLOGMEMBERS 5
    MAXLOGHISTORY 1
    MAXDATAFILES 100
    CHARACTER SET AL32UTF8
    NATIONAL CHARACTER SET AL16UTF16
    EXTENT MANAGEMENT LOCAL
    DATAFILE '/u01/oracle/oradata/ora11g/system01.dbf' SIZE 325M REUSE
    SYSAUX DATAFILE '/u01/oracle/oradata/ora11g/sysaux01.dbf' SIZE 325M REUSE
    DEFAULT TABLESPACE users
      DATAFILE '/u01/oracle/oradata/ora11g/users01.dbf'
      SIZE 500M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED
```

```
  DEFAULT TEMPORARY TABLESPACE tempts1
    TEMPFILE '/u01/oracle/oradata/ora11g/temp01.dbf'
    SIZE 20M REUSE
  UNDO TABLESPACE undotbs1
    DATAFILE '/u01/oracle/oradata/ora11g/undotbs01.dbf'
    SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

注意，上述内容中突出显示的部分即为修改的内容。

然后，回到第 5 步的窗口中，执行如下命令：

```
SQL> @/home/oracle/create_db.sql;
Database created.
```

7) 执行脚本。

```
SQL> @?/rdbms/admin/catalog.sql;
SQL> @?/rdbms/admin/catproc.sql;
SQL> conn system/oracle
Connected.
SQL> @?/sqlplus/admin/pupbld.sql
```

手工创建数据库完毕后，需要执行上述三个脚本。

8) 检查并设置 sqlplus 提示符

查看实例的状态：

```
SQL> select INSTANCE_NUMBER,INSTANCE_NAME,STATUS from v$instance;
INSTANCE_NUMBER INSTANCE_NAME    STATUS
--------------- ---------------- ------------
              1 ora11g           OPEN
1 row selected.
```

查看数据库的状态：

```
SQL> select DBID,NAME,OPEN_MODE from v$database;
     DBID  NAME       OPEN_MODE
---------- --------- --------------------
  11065102  ORA11G    READ WRITE
1 row selected.
```

接下来，完成数据库创建的最后一步：设置 sqlplus 提示符。

```
[oracle@rhel6 ~]$ vi $ORACLE_HOME/sqlplus/admin/glogin.sql
```

在该文件末尾处添加如下内容：

```
set sqlprompt "_user'@'_connect_identifier> "
```

然后保存退出。重新登录 sqlplus：

```
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
[oracle@rhel6 dbs]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Fri Apr 22 11:27:05 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SYS@ora11g>
```

至此，手工创建数据库的实验完成。

2.2 本章涉及的相关概念

数据库与实例

这两个概念应该是读者刚接触数据库技术时最容易混淆的概念了，而它们恰恰也是数据库技术最基本的两个概念。我们知道，Oracle 数据库是一个软件，在这个软件中，实例和数据库又分别指什么呢？

具备计算机基础的读者都知道，一个软件要运行，首先得有程序(也就是大段大段的代码)，有存放程序的文件。然后，运行的时候，会启动进程，会使用内存。在一个 Oracle 数据库中，所有进程和使用的内存就是实例(instance)。换言之，它是数据库启动之后才有的东西。而存储代码和程序以及其他数据的文件集合就是数据库(database)。简而言之，数据库就是存储在磁盘上的可见文件的集合。

每次使用数据库时，我们都要先启动它，这里的启动其实是指启动数据库的实例。如下：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Fri Apr 22 14:09:20 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SYS@ora11g> startup
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size          2257840 bytes
```

```
Variable Size       503319632 bytes
Database Buffers    327155712 bytes
Redo Buffers        2371584 bytes
Database mounted.
Database opened.
```

然后，当完成操作后，我们还需要关闭数据库，如下：

```
SYS@ora11g> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SYS@ora11g> exit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

pfile 与 spfile

这两个文件都是数据库的初始化参数文件。pfile 也就是 parameter file，spfile 则是 server-side parameter file。前者为文本格式的文件，我们可以直接编辑，后者则为二进制文件，无法直接手工编辑或修改。初始化参数文件中记录了数据库启动时所需的相关配置，是数据库实例启动时需要使用的第一个文件。当 pfile 和 spfile 均存在时，数据库默认使用 spfile；否则使用 pfile。在后面的控制文件多路复用实验中，我们将讲述如何利用这两个文件来修改初始化参数。

2.3　本章用到的 Linux 命令

本书并不太多关注 Linux 的相关知识，只就用到的 Linux 命名进行简单说明。

cd　切换目录。

ls　查看当前目录下的文件和子目录。

mkdir　make directory，创建目录。

pwd　显示当前所在目录。

cat　显示文件内容。

|　管道符常用于多个命令连接时，将前一个命令的输出作为后一个命令的输入。

grep　进行过滤操作。

vi　Linux 中最常用的文本编辑工具，需要掌握 vi 中常用的 i、a、o 等命令的用法。



第 3 章

SQL 基础系列实验

SQL 即结构化查询语言(Structured Query Language)，是管理和操作数据库最常用的一门语言。使用 SQL，我们可以完成数据库中各个对象的创建与管理，例如创建表、索引、视图等；也可以对数据库进行管理，例如创建表空间和删除数据文件等；还可以执行权限分配与收回，以及事务控制等操作。

SQL 语言基本可以分为如下 4 类：

1. DML：Data Manipulation Language

数据操作语言，我们常说的增(insert)、删(delete)、查(select)、改(update)就是 DML 语言。

2. DDL：Data Definition Language

数据定义语言，主要用来完成数据库对象的创建和管理，例如 create table、alter index 等。

3. DCL：Data Control Language

数据库控制语言，主要用于权限管理，例如 grant select on table1 to user1、revoke select on table1 from user1 等。

4. TCL：Transaction Control Statement

事务控制语言，主要用来控制事务的提交和回滚，例如 commit、rollback 等。

先来看一些简单例子。

连接到数据库：

```
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Fri Apr 22 14:32:11 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SYS@ora11g>
```

注意，这里是数据库已经启动后的情形。

当前使用的用户为 sys，我们通过上面的阴影着重显示部分可了解这一点。接下来，我们进行用户切换，我们要使用 scott 用户：

```
SYS@ora11g> conn scott/tiger;
ERROR:
ORA-01017: invalid username/password; logon denied
Warning: You are no longer connected to ORACLE.
@>
```

报错了，我们看一下这里的错误提示：用户名或密码无效，因此连接被拒绝。conn 为 connect 的缩写。

既然如此，我们去看一下 scott 用户，看看到底是怎么回事：

```
@> conn / as sysdba
```

```
Connected.
SYS@ora11g> select username,account_status from dba_users where
username='SCOTT';
no rows selected
```

我们这里使用 sys 用户查看 dba_users 表，结果发现数据库中没有这个用户。这是因为我们采用的是手工创建数据库的方式，数据库提供的测试用户和数据都不包含在新创建的数据库内。如果用 dbca 建库的话，可以勾选这些用户和数据。dba_users 这样的对象被称为数据字典表，里面记录了当前数据库中的所有用户及其相关信息。

既然如此，我们就利用 Oracle 提供的脚本来重建这个测试用户吧！

```
SYS@ora11g> @?/rdbms/admin/utlsampl.sql;
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

重新试一下：

```
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Fri Apr 22 14:43:16 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g>
```

完成该步骤后，就可以试着执行一些简单的 SQL 语句了。

3.1　简单 SQL 语句实验

查看当前用户拥有哪些表：

```
SCOTT@ora11g> select * from tab;
TNAME          TABTYPE CLUSTERID
------------------------------------------------
BONUS          TABLE
DEPT           TABLE
EMP            TABLE
SALGRADE       TABLE
```

查看 emp 表有哪些列，每列都是什么数据类型：

```
SCOTT@ora11g> desc emp;
 Name            Null?     Type
 ------------ --------- ----------------
 EMPNO           NOT NULL  NUMBER(4)
 ENAME                     VARCHAR2(10)
 JOB                       VARCHAR2(9)
 MGR                       NUMBER(4)
 HIREDATE                  DATE
 SAL                       NUMBER(7,2)
 COMM                      NUMBER(7,2)
 DEPTNO                    NUMBER(2)
```

注意，这里的 desc 为 describe 的缩写。
查看 emp 表中的全部数据：

```
SCOTT@ora11g> set line 200
SCOTT@ora11g> set pages 100
SCOTT@ora11g> select * from emp;
    EMPNO ENAME      JOB         MGR HIREDATE     SAL      COMM    DEPTNO
---------- ------------------- ------------------------ ------ ---------
      7369 SMITH      CLERK      7902 17-DEC-80    800       20
      7499 ALLEN      SALESMAN   7698 20-FEB-81   1600      300    30
      7521 WARD       SALESMAN   7698 22-FEB-81   1250      500    30
      7566 JONES      MANAGER    7839 02-APR-81   2975       20
      7654 MARTIN     SALESMAN   7698 28-SEP-81   1250     1400    30
      7698 BLAKE      MANAGER    7839 01-MAY-81   2850       30
      7782 CLARK      MANAGER    7839 09-JUN-81   2450       10
      7788 SCOTT      ANALYST    7566 19-APR-87   3000       20
      7839 KING       PRESIDENT       17-NOV-81   5000       10
      7844 TURNER     SALESMAN   7698 08-SEP-81   1500        0    30
      7876 ADAMS      CLERK      7788 23-MAY-87   1100       20
      7900 JAMES      CLERK      7698 03-DEC-81    950       30
      7902 FORD       ANALYST    7566 03-DEC-81   3000       20
      7934 MILLER     CLERK      7782 23-JAN-82   1300       10

14 rows selected.
```

查看 emp 表中共有多少行数据：

```
SCOTT@ora11g> select count(*) from emp;
```

```
  COUNT(*)
----------
 14
```

我们注意到，表 emp 的 deptno 列中有不少重复的值。如果我们想查看该列中不重复的数据，可使用如下命令：

```
SCOTT@ora11g> select distinct deptno from emp;
    DEPTNO
----------
 30
 20
 10
```

可见，该列只有三个不同的值。还可以对这个结果进行排序，如下：

```
SCOTT@ora11g> select distinct deptno from emp order by deptno;
    DEPTNO
----------
 10
 20
 30
```

这里的 order by 表示要按照 deptno 列进行排序，默认为升序。

3.2　表的创建与数据过滤实验

Oracle 数据库中，表是最重要的对象，用来存储数据。我们可以使用两种方式来创建表。
第一种方式：

```
SCOTT@ora11g> create table tab_test (id number,
name varchar2(30),
addr varchar2(50),
birth_day date);
Table created.
```

这种方式下，我们显式指定表名、表中包含的列以及列的数据类型。
第二种方式：

```
SCOTT@ora11g>create table emp_bakas select * from emp;
Table created.
```

这种方式称为参考建表，也就是参考另一个表的数据来创建我们所需要的表。这种方式也称为 CTAS 方式(注意上面语句的阴影加重部分)。

接下来，我们以刚创建的 emp_bak 表为例，来演示如何获取我们想要的数据。该表为员工表，里面存储了员工的编号、姓名、工作岗位、经理的编号、入职日期、工资、奖金系数以及部门编号等信息。

假如我们想知道部门编号为 10 的部门有哪些员工，可执行如下查询：

```
SCOTT@ora11g> select * from emp_bak where deptno = 10;
    EMPNO ENAME     JOB         MGR HIREDATE  SAL        COMM     DEPTNO
--------- -------- ---------- -------------- ---------- ------- ----------
     7782 CLARK    MANAGER    7839           09-JUN-81  2450     10
     7839 KING     PRESIDENT                 17-NOV-81  5000     10
     7934 MILLER    CLERK      7782           23-JAN-82  1300     10
```

这里引入 where 子句，也就是在前面的 select 语句的基础之上，我们添加一个过滤条件。如果我们想查询部门编号为 10 的部门里，哪些员工的工资超过 3000，查询如下：

```
SCOTT@ora11g> select * from emp_bak where deptno = 10 and sal > 3000;
    EMPNO ENAME     JOB         MGR HIREDATE  SAL        COMM     DEPTNO
--------- -------- ---------  -------------- ---------- -------- ---------
     7839 KING      PRESIDENT                17-NOV-81  5000     10
```

这条 SQL 语句是我们在上一条语句的基础之上，再添加一个过滤条件得到的。也是就是部门编号为 10，以及工资大于 3000 这两个条件同时满足，才是我们想要的数据。如果我们想知道销售人员和经理之间，有哪些人的工资高于 1500 呢？查询如下：

```
SCOTT@ora11g> select * from emp_bak where (job = 'SALESMAN' or job = 'MANAGER')
and sal > 1500;
    EMPNO ENAME     JOB         MGR HIREDATE    SAL      COMM     DEPTNO
--------- ------------------  --------------- -------- -------- ---------
     7499 ALLEN    SALESMAN   7698 20-FEB-81  1600     300       30
     7566 JONES    MANAGER    7839 02-APR-81  2975      20
     7698 BLAKE    MANAGER    7839 01-MAY-81  2850      30
     7782 CLARK    MANAGER    7839 09-JUN-81  2450      10
```

注意，本条 SQL 语句中，where 子句后面出现了 or 和 and。

or，表示两个条件成立一个即可。

and，则表示两个条件必须同时成立。

上述语句也可以这样写：

```
SCOTT@ora11g> select * from emp_bak where job in ('SALESMAN','MANAGER') and sal
> 1500;
```

如果想知道哪些员工的工资在 2500 和 3500 之间，SQL 语句如下：

```
SCOTT@ora11g> select empno,ename from emp_bak where sal between 2500 and 3500;
     EMPNO ENAME
---------- ----------
      7566 JONES
      7698 BLAKE
      7788 SCOTT
      7902 FORD
```

也可以这样写：

```
SCOTT@ora11g> select empno,ename from emp_bak where sal >= 2500  and sal <= 3500;
```

再看如下需求：

emp_bak 表中，ename 列记录了员工的名字。我们想知道哪些员工的名字以字母 A 开头，以及哪些员工的名字中的第二个字母是 A，SQL 如下：

```
SCOTT@ora11g> select empno,ename from emp_bak where ename like 'A%' or ename
like '_A%';
     EMPNO ENAME
---------- ----------
      7499 ALLEN
      7521 WARD
      7654 MARTIN
      7876 ADAMS
      7900 JAMES
```

这里我们在 where 子句的后面使用 like 来进行字符串匹配。其中，%表示匹配一个或多个字符，_表示只匹配一个字符。那么问题来了，如果我们想匹配的字符串中本身就包含_或%，该如何处理？

使用先前提到的 CTAS 方式创建一个表：

```
SCOTT@ora11g> create table tab_obj as select * from dba_objects;
create table tab_obj as select * from dba_objects
                                      *
ERROR at line 1:
ORA-00942: table or view does not exist
```

这里想使用 dba_objects 这个表来创建自己的表。结果数据库报错，说这个表不存在。其实这个表是存在的，它在 sys 用户名下。只不过我们当前的 scott 用户没有权限去访问它。那我们就为其授权：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> grant select on dba_objects to scott;
Grant succeeded.
```

我们切换到 sys 用户下，将对 dba_objects 的查询权限授予 scott 用户。然后继续：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create table tab_obj as select * from dba_objects;
Table created.
```

这样就可以了。然后我们利用刚创建的该表来进行一些查询。注意，该表中的数据较多，所以不要执行 select * from tab_obj。

dba_objects 也是一个数据字典表，它记录了当前数据库中所有对象的信息，包括表、视图等。我们继续前面的 like 查询，如果想查看当前数据库中共有多少对象的名称中包含'_'符号，可以执行如下查询：

```
SCOTT@ora11g> select count(object_name) from tab_obj where object_name like
'%\_%' escape '\';
COUNT(OBJECT_NAME)
------------------
     12197
```

显然，这里就遇到了我们前面提到的问题。也就是如果对象名中本身包含下划线，该如何过滤？注意上面 SQL 语句中的 escape，它表示反斜杠后面的下划线为要匹配的字符串中的内容，而非匹配一个字符。

关于 SQL 语句中的 where 子句，还有一个地方：null 值过滤。回到前面创建的 emp_bak，该表中的 comm 列为奖金。但很显然，不是所有人都有奖金，因此这一列中有些行没有值。如下：

```
SCOTT@ora11g> select distinct comm from emp_bak;
     COMM
----------

     1400
      500
      300
  0
```

可见，该查询语句返回的结果中，第一行是没有任何值的。在 Oracle 中，我们说这样的行包含 null 值。null 在数据库中的含义比较特殊，它称为“未知”。因此判断一行是否为 null 也

比较特殊：

```
SCOTT@ora11g> select  sal,comm from emp_bak where comm is null;
       SAL  COMM
---------- ----------
       800
      2975
      2850
      2450
      3000
      5000
      1100
       950
      3000
      1300

10 rows selected.
```

反之：

```
SCOTT@ora11g> select sal,comm from emp_bak where comm is not null;
       SAL  COMM
---------- ----------
      1600   300
      1250   500
      1250  1400
      1500     0
```

也就是说只能用 is null 或 is not null 来判断是否为空，而不能用= null：

```
SCOTT@ora11g> select  sal,comm from emp_bak where comm = null;
no rows selected
```

另外，null 和 null 也无法比较：

```
SCOTT@ora11g> select 1 from dual where null = null;
no rows selected

SCOTT@ora11g> select 1 from dual where null <> null;
no rows selected
```

这里的 dual 是 Oracle 数据库提供的一个对象，它只有一行一列：

```
SCOTT@ora11g> desc dual;
```

```
 Name                     Null?    Type
 -------------------- -------- ---------------------------------------------------
 DUMMY                             VARCHAR2(1)
SCOTT@ora11g> select * from dual;
D
-
X
```

3.3 基本函数应用实验

Oracle 中的函数分为单行函数和多行函数。其中，单行函数又分为字符函数、日期函数、数字函数、转换函数、通用函数以及分支函数。这些函数是用来进行数据处理的基本函数，也是在实际工作中会频繁用到的函数。

3.3.1 字符函数

字符函数是用来处理字符或字符串(以及其他可以转换为字符串的对象)的函数，包含如下函数：

lower() 小写转换函数

upper() 大写转换函数

initcap() 首字母大写函数

concat() 字符拼接函数

substr() 子串截取函数

length() 获取长度函数

instr() 子串位置查找函数

lpad() | rpad() 字符串向左/向右扩展函数

trim() 截取函数

replace() 替代函数

translate() 转换函数

我们来看如下例子：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Wed Apr 27 10:17:44 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SYS@ora11g> startup
ORACLE instance started.
Total System Global Area  835104768 bytes
```

```
Fixed Size      2257840 bytes
Variable Size    503319632 bytes
Database Buffers  327155712 bytes
Redo Buffers     2371584 bytes
Database mounted.
Database opened.
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> SELECT lower('MR.SHI') low,upper(' your name is ')
upp,initcap('yoRk mac mulin') init from dual;
LOW     UPP             INIT
------ -------------- --------------
mr.shi YOUR NAME IS  York Mac Mulin
```

上述三个函数是字符串中的大小写处理函数。lower 函数把输入的字母全部转换为小写，upper 函数把输入的字母全部转换为大写，而 initcap 函数则把输入的字母按照单词进行处理，将每个单词的首字母转换为大写。

再来看其他函数：

```
SCOTT@ora11g> select concat('your name is ','york') name from dual;
NAME
-----------------
your name is York
```

concat 函数用于将两个字符串进行拼接，然后合并输出。但是该函数一次只能拼接两个字符串。如果要将多个字符串拼接到一起，例如 your name、is just a、short name 这三个字符串，那就需要使用 concat 的嵌套了。如下：concat(concat('your name', 'is just a'), 'short name')。显然这样就较复杂了。我们可以使用||拼接符进行处理。

```
SCOTT@ora11g> select 'your name '||'is just a '||'short name' name from dual;
NAME
------------------------------
your name is just a short name
```

目前，我们在 scott 用户下构建了几个测试表，如下：

```
SCOTT@ora11g> select * from tab;
TNAME         TABTYPE CLUSTERID
------------- ---------------------------------
BONUS         TABLE
DEPT          TABLE
EMP           TABLE
```

```
EMP_BAK        TABLE
SALGRADE       TABLE
TAB_OBJ        TABLE
TAB_TEST       TABLE
7 rows selected.
```

其中，带下划线的三张表是我们创建的测试表。如果我们想删除这三个表，该如何处理？这里，可使用||拼接出要删除这三张表的 DDL 语句。

```
SCOTT@ora11g> select 'drop table '||tname||' purge;' from tab where tname like
'%\_%' escape '\';
'DROPTABLE'||TNAME||'PURGE;'
------------------------------------------------
drop table EMP_BAK purge;
drop table TAB_OBJ purge;
drop table TAB_TEST purge;
```

此后，执行由查询结果拼接而成的 DDL 语句，就可以删除这些表了：

```
SCOTT@ora11g> drop table EMP_BAK purge;
Table dropped.
SCOTT@ora11g> drop table TAB_OBJ purge;
Table dropped.
SCOTT@ora11g> drop table TAB_TEST purge;
Table dropped.
```

这是||拼接符极有用的一个实用例子，尤其当我们想批量处理一些对象时，将可以这么做。下面继续讨论 substr 函数，这是一个相当常用的子串截取函数。我们来看例子：

```
SCOTT@ora11g> select substr('ABCDEFG',3,4) sub from dual;
SUB
----
CDEF
```

该函数中，第 1 个参数为要处理的字符串，第 2 个参数为开始截取的位置，第 3 个参数为截取的长度。上例就是要从 ABCDEFG 的第 3 个位置开始，截取接下来的 4 个字符。那么，我们可以稍微扩展一下，如果上例中第 2 个参数为 0 或负数，则返回结果又该是怎样的？

```
SCOTT@ora11g> select substr('ABCDEFG',-4,4) sub from dual;
SUB
----
DEFG
```

```
SCOTT@ora11g> select substr('ABCDEFG',0,4) sub from dual;
SUB
----
ABCD
```

length 函数用来获取字符串的长度：

```
SCOTT@ora11g> select length('CANDIDE') len from dual;
      LEN
----------
  7
```

instr 函数用于查找子串位置：

```
SCOTT@ora11g> select instr('coporate floor','or',3,2) instr from dual;
    INSTR
----------
 13
```

上述例子中，instr 函数中的第 1 个参数表示要处理的字符串；第 2 个参数表示要查找在第一个参数中出现位置的子串；第 3 个参数表示从第一个参数中的第几位开始查找；第 4 个参数则表示其第几次出现的位置。因此，该例演示的是：从第 3 位开始查找子串 or 在 coporate floor 中第 2 次出现的位置。

lpad 与 rpad 函数用来对字符串进行扩展：

```
SCOTT@ora11g> select lpad('test',12,'*') lp,rpad('test',12,'*') rp from dual;
LP           RP
------------ ------------
********test test********
```

其中，l 为 left，意味着向左扩展；r 为 right，意味着向右扩展。函数中的第 2 个参数为扩展后的字符串长度，第 3 个参数指示用哪些字符进行扩展。但要注意如下用法：

```
SCOTT@ora11g> select lpad('just_a_test',10,'*') lp from dual;
LP
----------
just_a_tes
```

这里，第 1 个参数的长度就已经超过要扩展后的长度了，因此该例实际上是字符串截取，它等同于如下方式：

```
SCOTT@ora11g> select substr('just_a_test',1,10) sub from dual;
SUB
```

```
----------
just_a_tes
```

也就是说，上面两种 SQL 写法，可以满足同样的需求。但在实际工作中，建议还是按照正向思维来处理。这样，无论对于开发者还是后期运维人员来说，代码都具有更好的可读性和可维护性。

trim 函数用来对字符串的左右端进行截取，例子如下：

```
SCOTT@ora11g> select trim ('  test  ') from dual;
TRIM
----
test
```

默认情况下，trim 函数会将待处理的字符串左右两端的空格截掉。但有时字符串左右两端可能不是空格，而且我们也可能只想截掉左端或右端的指定字符：

```
SCOTT@ora11g> select ltrim('<=====>BROWNING<=====>','<>=') lt,
rtrim('<=====>BROWNING<=====>','<>=') rt from dual;
LT  RT
--------------- ---------------
BROWNING<=====><=====>BROWNING
```

replace 函数用来使用指定的字符代替被处理的字符，例子如下：

```
SCOTT@ora11g> select replace ('Jack and Jue','J','B') rep from dual;
REP
--------------
BLack and Blue
```

也可用 translate 函数来实现：

```
SCOTT@ora11g> select translate ('Jack and Jue','J','B') rep from dual;
REP
------------
Back and Bue
```

字符函数可能是实际工作中用得最多的函数了，因此对于上述字符函数，建议初学者一定要认真掌握，并能熟练查阅官方文档。

3.3.2 日期函数

日期函数也是常用函数之一，主要包含如下函数：

months_between() 用于计算两个日期之间相差的月数

add_months()　在指定日期上进行月份添加
next_day()　指定日期的下一个日期
last_day()　指定日期所在月份的最后一天
round()　日期的四舍五入
trunc()　日期的截断
Oracle 数据库中，默认的日期类型值的显示格式为 DD-MON-RR：

```
SCOTT@ora11g> select sysdate from dual;
SYSDATE
---------
28-APR-16
```

这里的 sysdate 是系统提供的函数，用于获取操作系统的当前时间。按照数据库默认的时间格式，显示为日期-月份(英文的前三个字母)-年份(后两位)。但是这种格式往往不太实用。我们需要做一些调整，这个在后面的 3.3.5 一节中再做详细说明。

来看如下例子：

```
SCOTT@ora11g> select months_between(sysdate, sysdate - 100) mon from dual;
      MON
----------
3.29032258
```

months_between函数用于返回两个日期之间相差的月数。如果第二个参数大于第一个参数，则返回结果为正，否则为负。

```
SCOTT@ora11g> select add_months(sysdate,2) add_mon,
                add_months(sysdate,-2) sub_mon from dual;
ADD_MON   SUB_MON
--------- ---------
28-JUN-16 28-FEB-16
```

add_months 函数用于在指定日期上加上或减去一定的月份，如上例所示。

```
SCOTT@ora11g> select next_day(sysdate,'Fri') next from dual;
NEXT
---------
29-APR-16
```

next_day 用于返回指定日期的下一个星期几，上例中是返回下一个星期五。

```
SCOTT@ora11g> select last_day(sysdate) last from dual;
LAST
---------
```

```
30-APR-16
```

last_day 用于返回指定日期所在月份的最后一天。

Oracle 引入这些函数用于处理日期类型的值，原因也非常简单。日期类型的数值不同于二进制、十进制这样的数值。我们知道，要获取一年有多少天，不同的年份返回结果是不一样的。同样，一个月有多少天，结果也是不确定的。因此，提供这些函数，就可以处理日期这种进制不确定的数值了。

常用的日期函数还有 round 和 trunc，如下例：

```
SCOTT@ora11g> select round(sysdate+100,'year') year,
trunc(sysdate+100,'year') year1 from dual;

YEAR      YEAR1
--------- ---------
01-JAN-17 01-JAN-16
```

这两个函数，实际上就是对日期进行四舍五入或截取。

3.3.3 数字函数

常用的数字函数比较简单，如下：

round() 对指定数值进行四舍五入

trunc() 对指定数值进行截断

mod() 取余函数

来看例子：

```
SCOTT@ora11g> select round(123.55) rou,trunc(123.55) tru from dual;
       ROU   TRU
---------- ----------
       124   123
```

默认情况下，round 和 trunc 函数，都是对小数点后面的数值进行处理，要么四舍五入，要么截断。考虑再添加第二个参数的情形：

```
SCOTT@ora11g> select round(456.78,-2) rou,trunc(456.78,1) tru from dual;
       ROU   TRU
---------- ----------
        500 456.7
```

这里以小数点为界，第二个参数为正，则向右移动对应的位数，然后进行处理；第二个参数为负，则向左移动对应的位数，然后进行处理。

mod 函数则用来进行取余，或者称为取模函数。如下：

```
SCOTT@ora11g> select mod (100,3) mod from dual;
       MOD
----------
  1
```

上例中，我们取 100 除以 3 的余数，结果为 1。现在，利用 mod 函数，我们可以满足如下需求，我们想生成 0 和 100 之间的随机数，用 SQL 来实现：

```
SCOTT@ora11g> select abs(mod(dbms_random.random,101)) rand_val from dual;
  RAND_VAL
----------
 94
```

我们用到了简单的函数嵌套。先使用数据库提供的 dbms_random.random 生成随机数，然后取余，对其返回值的范围进行限制，然后取其绝对值，剔除负值。

3.3.4　通用函数

通用函数又称为 null 值处理函数，当然实际上这个称呼并不是太精确，主要包含如下 4 个函数：

```
nvl()
nvl2()
nullif()
coalesce()
```

我们先来看相关实验，然后总结其用途：

```
SCOTT@ora11g> set pages 100
SCOTT@ora11g> select comm from emp;
      COMM
----------

       300
       500

      1400





  0
```

```
14 rows selected.
```

我们在前面已经提到了 null 的概念。emp 表中，comm 列为奖金系数列，它乘以 sal 列，就是员工每个月应该拿到手的奖金。但是有些员工没有奖金，也就是 comm 列对应的值为 null。null 有一个非常有趣的特点就是，一旦它参与算术运算，无论是何种算术运算，结果都会为 null。因此，如果想计算所有员工的总收入，就需要处理 comm 为 null 的值，如下：

```
SCOTT@ora11g> select nvl(comm,0) from emp;
NVL(COMM,0)
-----------
   0
 300
 500
   0
        1400
   0
   0
   0
   0
   0
   0
   0
   0
   0

14 rows selected.
```

这里使用了 nvl 函数，其用法是，如果第一个参数为 null，则返回第二个参数的值，否则返回第一个参数。这里将 null 值转换为 0，这样就可以参与算术运算了：

```
SCOTT@ora11g> select empno,ename,sal,comm,sal + nvl(comm,0) total from emp;
    EMPNO ENAME   SAL    COMM   TOTAL
--------- ------- ------ ------ ----------
     7369 SMITH    800     800
     7499 ALLEN   1600     300  1900
     7521 WARD    1250     500  1750
     7566 JONES   2975    2975
```

```
      7654 MARTIN  1250      1400  2650
      7698 BLAKE   2850     2850
      7782 CLARK   2450     2450
      7788 SCOTT   3000     3000
      7839 KING    5000     5000
      7844 TURNER  1500        0  1500
      7876 ADAMS   1100     1100
      7900 JAMES    950      950
      7902 FORD    3000     3000
      7934 MILLER  1300     1300

14 rows selected.

SCOTT@ora11g> select empno,ename,sal,comm,nvl2(comm,sal + comm,sal) income
from emp;
     EMPNO ENAME   SAL     COMM   INCOME
--------- ------- ------- ------ ----------
      7369 SMITH    800      800
      7499 ALLEN   1600      300   1900
      7521 WARD    1250      500   1750
      7566 JONES   2975     2975
      7654 MARTIN  1250     1400   2650
      7698 BLAKE   2850     2850
      7782 CLARK   2450     2450
      7788 SCOTT   3000     3000
      7839 KING    5000     5000
      7844 TURNER  1500        0   1500
      7876 ADAMS   1100     1100
      7900 JAMES    950      950
      7902 FORD    3000     3000
      7934 MILLER  1300     1300

14 rows selected.
```

与 nvl 函数不同，nvl2 函数有三个输入参数，其含义为：如果第一个参数不为 null，则返回第二个参数的值，否则返回第三个参数。

```
SCOTT@ora11g> select nullif(length('test'),length('name')) val from dual;
       VAL
----------
```

严格来讲，nullif 函数并不是 null 处理函数。该函数的用法是，如果第一个参数和第二个参数相等，则返回 null。上述例子中，我们分别取字符串 test 和 name 的长度，结果这两个字符串的长度相同，因此该 SQL 返回结果为 null。

```
SCOTT@ora11g> select coalesce(null,null,5) val from dual;
       VAL
----------
  5
```

coalesce 函数则用于判断输入参数中第一个非 null 的参数。如果当前参数为 null，则继续往下判断，直到判断到一个非 null 的参数为止，并返回该参数。如果所有输入参数都为 null，则返回 null：

```
SCOTT@ora11g> select coalesce(null,null,null) val from dual;
V
-
```

3.3.5 转换函数

在 Oracle 数据库中，最常用的数据类型莫过于日期、字符以及数字三种。而有时，我们知道，这些类型是可以进行相互转换的。例如，1011 是一个字符串，但是显然也可以作为数字 1011 进行处理。再比如，2016-04-28 也是一个字符串，但是显然也可以表示一个日期。

数据库中，Oracle 能够完成一些数据类型之间的隐式转换，例如：

```
SCOTT@ora11g> desc emp;
 Name        Null?    Type
 ----------- -------- ----------------------------
 EMPNO       NOT NULL  NUMBER(4)
 ENAME                 VARCHAR2(10)
 JOB                   VARCHAR2(9)
 MGR                   NUMBER(4)
 HIREDATE              DATE
 SAL                   NUMBER(7,2)
 COMM                  NUMBER(7,2)
 DEPTNO                NUMBER(2)
```

查看 scott 用户下的 emp 表，我们可以知道 empno 列是数值类型。但用如下 SQL 照样可以执行：

```
SCOTT@ora11g> select empno,ename from emp where empno ='7900';

     EMPNO ENAME
```

```
---------- ----------
     7900 JAMES
```

这里，Oracle 就用到了隐式数据类型转换。它会将 empno 的值取出，然后转换为字符类型，再与 7900 进行比较。但有时，Oracle 使用隐式数据类型转换可能导致索引失效，从而使 SQL 执行性能下降。因此，我们通常建议使用显式数据类型转换。也就是使用类型转换函数。关于 index(索引)，我们将在后文讨论。

Oracle 提供了三种数据类型转换函数：

to_char()

to_date()

to_number()

实验如下：

```
SCOTT@ora11g> select to_char(sysdate,'yyyy-mm-dd hh24:mi:ss') sys_date from dual;
SYS_DATE
-------------------
2016-04-28 15:53:00
SCOTT@ora11g> select to_char(sal,'$9999') sal from emp where empno = 7900;
SAL
------
  $950
```

上述两个例子使用 to_char 函数，分别将日期类型和数值类型的值转换为字符型。

```
SCOTT@ora11g> select to_date('January 15,1998','Month dd,YYYY') date_val from dual;
DATE_VAL
---------
15-JAN-98
```

上述示例将字符串转换为日期类型的值。

```
SCOTT@ora11g> select to_number('20160428') from dual;
TO_NUMBER('20160428')
---------------------
     20160428
```

上述示例将字符串转换为数字。

需要注意，日期类型和数值类型无法直接转换，需要先转换为字符类型，然后转换为数值类型或日期类型。

3.3.6 分支函数

开发人员都知道，我们有一种语法用来进行条件判断：if-then-else。Oracle 在 SQL 中也实现了类似功能。使用的函数是 case 和 decode，如下：

```
SCOTT@ora11g> select empno,ename,sal,
                case when sal <= 3000 then 'low'
                    when sal > 3000 and sal <= 5000 then 'mid'
                    else 'high'
                end sal_level
          from emp;
    EMPNO ENAME   SAL  SAL_
--------- ------- ---- ---------
     7369 SMITH    800 low
     7499 ALLEN   1600 low
     7521 WARD    1250 low
     7566 JONES   2975 low
     7654 MARTIN  1250 low
     7698 BLAKE   2850 low
     7782 CLARK   2450 low
     7788 SCOTT   3000 low
     7839 KING    5000 mid
     7844 TURNER  1500 low
     7876 ADAMS   1100 low
     7900 JAMES    950 low
     7902 FORD    3000 low
     7934 MILLER  1300 low

14 rows selected.
```

上述语句中，我们对 sal 进行判断，若工资小于等于 3000，则返回其工资级别为 low；若在 3000 至 5000 之间，则返回 mid，否则返回 high。

再来看 decode 函数的实验：

```
SCOTT@ora11g> select empno,ename,sal,
                decode(trunc(sal/2000,0),
                        0,0,
                        1,0.1,
                        2,0.2,
                          0.3) tax_rate
          from emp;
```

```
     EMPNO ENAME     SAL    TAX_RATE
---------- ------------- ----------
      7369 SMITH      800    0
      7499 ALLEN     1600    0
      7521 WARD      1250    0
      7566 JONES     2975   .1
      7654 MARTIN    1250    0
      7698 BLAKE     2850   .1
      7782 CLARK     2450   .1
      7788 SCOTT     3000   .1
      7839 KING      5000   .2
      7844 TURNER    1500    0
      7876 ADAMS     1100    0
      7900 JAMES      950    0
      7902 FORD      3000   .1
      7934 MILLER    1300    0

14 rows selected.
```

这里使用 decode 函数来判断员工的工资，查看其纳税比率。如果工资低于 2000，则不纳税(税率为 0)；如果工资在 2000 到 4000 之间，则纳税比率为 10%；如果在 4000 到 6000 之间，则纳税比率为 20%；再高，则为 30%。

3.4　组函数练习实验

多行函数又称为组函数。我们前面所演示的函数例子，都基于数据一条一条地进行处理。但组函数却基于一组数据进行处理，每组数据返回一个结果。这个组可以是一个表，也可以是一个表的一部分数据。常用的组函数如下：

avg()　求平均值
count()　求总数
max()　求最大值
min()　求最小值
sum()　求和

这些函数的意义都比较简单，我们来看相应的示例：

```
SCOTT@ora11g> select avg(sal),count(*),min(sal),max(sal),sum(sal) from emp;
  AVG(SAL)   COUNT(*)   MIN(SAL)   MAX(SAL)   SUM(SAL)
---------- ---------- ---------- ---------- ----------
2073.21429         14        800       5000      29025
```

我们这里是把 emp 整个表当成一个组，然后查看所有员工的平均工资、员工总数、最低工资、最高工资以及工资总额。

如果我们想知道每个部门的平均工资，就需要根据部门进行分组，然后求平均值：

```
SCOTT@ora11g> select deptno,avg(sal)
from emp
group by deptno;
    DEPTNO   AVG(SAL)
---------- ----------
 30 1566.66667
 20  2175
 10 2916.66667
```

这里新引入了 group by 关键字，也就是按哪些列进行分组。如果我们还想在分组的基础上再进行过滤呢？例如，我们想查看一下哪些部门的平均工资超过 2000：

```
SCOTT@ora11g> select deptno,avg(sal)
from emp
group by deptno
having avg(sal) > 2000;
    DEPTNO   AVG(SAL)
---------- ----------
 20         2175
 10         2916.66667
```

这里的 having 关键字，则是在 group by 的基础之上再进行过滤。也就是说，只有使用了 group by，才能使用 having 子句。

这里稍微补充一下，如果我们还想对 SQL 执行结果进行排序呢？例如上述例子中，我们想按照部门编号进行升序排列：

```
SCOTT@ora11g> select deptno,avg(sal)
from emp
group by deptno
having avg(sal) > 2000
order by deptno;
    DEPTNO   AVG(SAL)
---------- ----------
 10         2916.66667
 20         2175
```

3.5　DML 操作实验

前面在讲 SQL 分类时，我们提到 DML 操作包含的几种常见操作。实际上，除了 select 用来进行数据查询之外，insert、update 和 delete 都用来进行数据修改操作。我们先创建一个测试表：

```
SCOTT@ora11g> create table tab_test (
                 id number,
                 name varchar2(30));
Table created.
```

然后进行简单的数据插入：

```
SCOTT@ora11g> insert into tab_test values (1,2);
1 row created.
SCOTT@ora11g> commit;
Commit complete.
```

但是这样的插入，一次只能插入一条记录。我们可以使用循环插入，一次性插入 10 条记录：

```
SCOTT@ora11g> begin
   for i in 1..10 loop
   insert into tab_test values (i,'test'||i);
   end loop;
   commit;
   end;
   /
PL/SQL procedure successfully completed.
```

当然，也可以用其他方式插入多条记录，如下：

```
SCOTT@ora11g> insert into tab_test select * from tab_test;
11 rows created.
SCOTT@ora11g> commit;
Commit complete.
```

这样，我们使用子查询的方式来插入数据，一次就可以插入多条记录了。

然后修改数据：

```
SCOTT@ora11g> update tab_test set id = 2 where id =1;
4 rows updated.
```

```
SCOTT@ora11g> commit;
Commit complete.
```

指定 where 条件，我们可以更新满足 where 过滤条件的记录。如果不指定 where 条件，则更新所有数据：

```
SCOTT@ora11g> update tab_test set id = 100;
22 rows updated.
SCOTT@ora11g> commit;
Commit complete.
```

删除数据：

```
SCOTT@ora11g> delete from tab_test where name = 'test2';
2 rows deleted.
SCOTT@ora11g> commit;
Commit complete.
```

与 update 一样，指定 where 条件，则删除满足 where 条件的记录，不指定 where 条件，则删除表中所有记录：

```
SCOTT@ora11g> delete from tab_test;
20 rows deleted.
SCOTT@ora11g> commit;
Commit complete.
```

实际上，还有一种删除全表数据的方法：

```
SCOTT@ora11g> truncate table tab_test;
Table truncated.
```

truncate 其实是 DDL 语句，与 delete 相比，它速度更快。但一旦出现数据删除错误，恢复起来也比较麻烦，因此要慎用。除非你确定表中的数据将来肯定不会再用，才可以使用 truncate。

3.6 其他数据库对象创建与管理实验

数据库中除表之外，还有索引、视图、序列、同义词、临时表等多种对象。本节将演示如何创建这些对象并对其进行管理。

索引(index)

Oracle 引入索引的目的只有一个，就是要加快查询的执行速度。前面演示的各种实验的数据量都较小，SQL 语句也都比较简单，因此 SQL 的执行速度不成问题。但如果在生产系统中，

我们要在一个包含上千万甚至上亿条记录的表中查询数据，那么速度可能就没有那么快了。分析下例：

```
SCOTT@ora11g> create table tab_obj as select * from dba_objects;
Table created.
SCOTT@ora11g> set timing on;
SCOTT@ora11g> select count(*) from tab_obj;
  COUNT(*)
----------
     13524
Elapsed: 00:00:00.01
SCOTT@ora11g> set timing off;
```

当创建的表中只有 13 524 条记录时，该 SQL 的执行时间仅需 0.01 秒。这里的 set timing 语句用来设置当前 sqlplus 的计时功能。将其设置为 on 时，我们就可以查看接下来的操作中每一步具体都消耗了多长时间。

接下来重复插入数据，使得 tab_obj 中的数据量变大：

```
SCOTT@ora11g> insert into tab_obj select * from tab_obj;
13524 rows created.
SCOTT@ora11g> /
27048 rows created.
SCOTT@ora11g> /
54096 rows created.
SCOTT@ora11g> /
108192 rows created.
SCOTT@ora11g> /
216384 rows created.
SCOTT@ora11g> /
432768 rows created.
SCOTT@ora11g> /
865536 rows created.

SCOTT@ora11g> set timing on;
SCOTT@ora11g> select count(*) from tab_obj;
  COUNT(*)
----------
   1731072
Elapsed: 00:00:00.10
SCOTT@ora11g> set timing off;
```

当数据量增大到 1 731 072 时，同样的 SQL 语句，其执行时间已经变成 0.1 秒。如果数据量继续增加，则执行时间还会增加。而实际上，随着现在业务系统中数据量的不断增加，数据库中有包含上亿条记录的表已是常见的事情了。

当我们执行 SQL 来查询数据时，例如有这样的 SQL：

```
select * from tab_obj where object_id > 9527 and object_type='TABLE';
```

如果没有索引，为完成本条 SQL，数据库只能对 tab_obj 中的所有记录一条一条地按照 where 过滤条件进行校验。如果满足 where 条件，则返回；否则抛弃。但是一旦有了索引并且可以使用该索引来获得本 SQL 所需要的数据，数据库就不需要去逐行扫描 tab_obj 中的所有记录了(这种数据访问方式称为全表扫描)，而是去索引中查找对应的记录就行了。索引中的数据是有序存放的，并且每条记录和其实际的存储位置是对应的。这就如我们翻书时，按目录去查找对应的页码一样，速度就会快很多。

本书侧重关注 Oracle 数据库的快速上手实验，因此并不深入探讨索引的结构及原理。相关内容会在后续书籍中进行详细阐述。

来看如何创建索引：

```
SCOTT@ora11g> create index ind_id on tab_obj(object_id);
Index created.
```

索引主要分为两类：B*Tree 索引和 Bitmap 索引。它们有各自的适用场景。创建 B*Tree 索引就是使用上述方式，括号中为列名。当然也可以是列的组合，或者是基于列的表达式。前者称为复合索引，后者称为函数索引，如下：

```
SCOTT@ora11g> create index ind_comp on tab_obj(object_id,object_name);
create index ind_comp on tab_obj(object_id,object_name)
                            *
ERROR at line 1:
ORA-01652: unable to extend temp segment by 128 in tablespace TEMPTS1
```

这里报告 01652 错误，我们打开一个新窗口，看一下是怎么回事：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Fri Apr 29 09:16:46 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SYS@ora11g> desc dba_temp_files;
 Name                 Null?     Type
```

```
 ---------------- -------- ----------------------------
 FILE_NAME                 VARCHAR2(513)
 FILE_ID                   NUMBER
 TABLESPACE_NAME  NOT NULL VARCHAR2(30)
 BYTES                     NUMBER
 BLOCKS                    NUMBER
 STATUS                    VARCHAR2(7)
 RELATIVE_FNO              NUMBER
 AUTOEXTENSIBLE            VARCHAR2(3)
 MAXBYTES                  NUMBER
 MAXBLOCKS                 NUMBER
 INCREMENT_BY              NUMBER
 USER_BYTES                NUMBER
 USER_BLOCKS               NUMBER
SYS@ora11g> select BYTES/1024/1024 from dba_temp_files;
BYTES/1024/1024
---------------
     20
```

此处是因为创建索引时需要使用临时表空间，而我们初始的临时表空间太小，所以对它进行调整。打开临时表空间中临时文件的自动扩展，最大值设置为 10GB。关于临时表空间的概念，会在后面的 4.4 一节中进行讲解。

```
SYS@ora11g> alter database tempfile 1 autoextend on maxsize 10G;
Database altered.
```

然后回到原窗口，创建复合索引：

```
SCOTT@ora11g> create index ind_comp on tab_obj(object_id,object_name);
Index created.
```

我们再来创建一个函数索引：

```
SCOTT@ora11g> create index ind_func on tab_obj(substr(object_name,1,5));
Index created.
```

再来看创建 Bitmap 索引的写法：

```
SCOTT@ora11g> create bitmap index ind_bit on tab_obj(object_type);
Index created.
```

创建完索引后，我们会在第 6 章的相关实验中详细讲解如何使用索引，这里暂不做描述。

索引创建完毕后，Oracle 会自动对其进行维护。当索引所在表中的数据发生变化时，索引

中的内容也会随之变化。这里需要注意的是，函数索引是对列进行函数运算，然后保存结果。也就是说，每次表中数据发生变化时，函数索引的值都将重新计算(当然，是索引中的列在表内的数据发生了变化。例如上例中 object_type 列的值发生变化，索引 ind_func 的值才会重新计算)。因此，如果我们想使用函数索引，则该函数千万不要太复杂，否则 Oracle 数据库维护索引的成本会较高。

视图(view)

为完成数据查询，很多时候我们需要编写很长的 select 语句，有时甚至达到上百行代码。那么我们是否可将这些代码存储下来，然后给它指定一个别名，以后再执行这些代码的时候，直接使用别名呢？

视图就用来完成这样的事情。当然它的用途还不限于此。使用视图最大的好处，实际上可能是能够对用户屏蔽底层代码实现，使得应用程序对终端用户透明。比如我们只让用户使用一个视图，而该视图的语法，具体使用了哪些表，这些表的结构如何，用户都无从知晓。

下面创建简单的视图：

```
SCOTT@ora11g> create view v_test as select * from emp where deptno > 10;
create view v_test as select * from emp where deptno > 10
           *
ERROR at line 1:
ORA-01031: insufficient privileges
```

这里报告了 01031 错误，指当前的 scott 用户没有创建视图的权限，我们需要切换到 sys 用户，然后对 scott 用户进行授权。这里的 grant 语句就是前面提到的 SQL 分类中的 DCL 语句。

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> grant create view to scott;
Grant succeeded.
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create view v_test as select * from emp where deptno > 10;
View created.
```

这样，以后再想查询 emp 表中哪些部门的编号大于 10 时，就可以这么做：

```
SCOTT@ora11g> select * from v_test;
    EMPNO ENAME      JOB         MGR HIREDATE         SAL      COMM  DEPTNO
--------- ---------- --------- ----------------- -------- ------- ----------
     7369 SMITH      CLERK       7902 17-DEC-80    800      20
     7499 ALLEN      SALESMAN    7698 20-FEB-81   1600      300    30
     7521 WARD       SALESMAN    7698 22-FEB-81   1250      500    30
```

```
     7566 JONES      MANAGER     7839 02-APR-81   2975       20
     7654 MARTIN      SALESMAN   7698 28-SEP-81   1250    1400    30
     7698 BLAKE      MANAGER     7839 01-MAY-81   2850       30
     7788 SCOTT      ANALYST     7566 19-APR-87   3000       20
     7844 TURNER      SALESMAN   7698 08-SEP-81   1500        0    30
     7876 ADAMS      CLERK       7788 23-MAY-87   1100       20
     7900 JAMES      CLERK       7698 03-DEC-81    950       30
     7902 FORD       ANALYST     7566 03-DEC-81   3000       20
11 rows selected.
```

此时，查询该视图，就相当于重新运行 select * from emp where deptno > 10。当然这里还有一个遗留问题。我们只是把 select 语句存储了下来，每次查询该视图的时候，还是需要去运行相应的 select 语句。借助物化视图，我们可以把该视图对应的 select 语句的查询结果直接保存下来，以后再查询该视图的时候，就不用去执行 select 了。它将 select 语句的查询结果以表的形式保存下来。

序列(sequence)

有时，我们想把一张表中的每条记录都用唯一编号进行标识，例如使用 1、2、3、4、5…这样的数字。Oracle 提供了一个可自动生成并供我们使用的小对象，这个对象就是序列。其创建方式如下：

```
SCOTT@ora11g> create sequence seq_test
start with 1
increment by 1
maxvalue 9999
nocycle
cache 20;
Sequence created.
```

本例中，start with 表示该序列的初始值；increment by 表示步长，也就是每次增加多少；maxvalue 表示该序列的最大值；nocycle 表示增长到最大值后就不再循环；cache 表示将该序列接下来的 20 个值缓存到内存中。

创建完序列后，来看如何获取该序列中的值。这里要用到两个函数 currval 和 nextval，前者获取序列的当前值，后者获取序列的下一个值。

```
SCOTT@ora11g> select seq_test.currval from dual;
select seq_test.currval from dual
      *
ERROR at line 1:
ORA-08002: sequence SEQ_TEST.CURRVAL is not yet defined in this session
```

需要注意，创建完序列后，第一次使用序列时，不能使用 currval 函数，而要使用 nextval 函数：

```
SCOTT@ora11g> select seq_test.nextval from dual;
   NEXTVAL
----------
  1
SCOTT@ora11g> /
   NEXTVAL
----------
  2
SCOTT@ora11g> /
   NEXTVAL
----------
  3
SCOTT@ora11g> /
   NEXTVAL
----------
  4
SCOTT@ora11g> select seq_test.currval from dual;
   CURRVAL
----------
  4
```

前面曾创建一个测试表 tab_test，现在就使用序列为其插入数据：

```
SCOTT@ora11g> desc tab_test;
 Name               Null?    Type
 ----------------------- ------------------------------
 ID                          NUMBER
 NAME                        VARCHAR2(30)
SCOTT@ora11g> insert into tab_test values (seq_test.nextval,'test001');
1 row created.
SCOTT@ora11g> commit;
Commit complete.
```

同义词(synonym)

同义词实际上就是数据库中原有对象的一个别名，使用同义词可以简化访问其他用户的对象。其创建语法很简单：

```
SCOTT@ora11g> create public synonym syn_emp for emp;
create public synonym syn_emp for emp
```

```
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

这里的错误与前面的错误一样，都是当前的 scott 用户缺乏创建同义词的权限。我们需要切换到 sys 用户进行授权：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> grant create public synonym to scott;
Grant succeeded.
```

然后切换到 scott 用户，为 emp 表创建同义词：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create public synonym syn_emp for emp;
Synonym created.
SCOTT@ora11g> grant select on syn_emp to public;
Grant succeeded.
```

接下来创建一个测试用户，并授予其基本连接数据库和创建对象的权限：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> create user u_test1 identified by test;
User created.
SYS@ora11g> grant resource,connect to u_test1;
Grant succeeded.
```

这里的 resource、connect 称为角色。

这样，我们就可以在 u_test1 用户下查看 syn_emp 了，当然，实际上查看的是 scott 用户下 emp 表中的内容。

```
SYS@ora11g> conn u_test1/test;
Connected.
U_TEST1@ora11g> select * from syn_emp;
    EMPNO ENAME  JOB        MGR HIREDATE       SAL     COMM  DEPTNO
--------- ------ --------- --------------- ------- -----  ----------
     7369 SMITH   CLERK     7902 17-DEC-80    800        20
     7499 ALLEN  SALESMAN  7698 20-FEB-81   1600      300  30
     7521 WARD    SALESMAN  7698 22-FEB-81   1250      500  30
     7566 JONES   MANAGER   7839 02-APR-81   2975        20
```

```
      7654 MARTIN  SALESMAN  7698 28-SEP-81   1250    1400  30
      7698 BLAKE   MANAGER   7839 01-MAY-81   2850      30
      7782 CLARK   MANAGER   7839 09-JUN-81   2450      10
      7788 SCOTT   ANALYST   7566 19-APR-87   3000      20
      7839 KING    PRESIDENT      17-NOV-81   5000      10
      7844 TURNER  SALESMAN  7698 08-SEP-81   1500       0  30
      7876 ADAMS   CLERK     7788 23-MAY-87   1100      20
      7900 JAMES   CLERK     7698 03-DEC-81    950      30
      7902 FORD    ANALYST   7566 03-DEC-81   3000      20
      7934 MILLER  CLERK     7782 23-JAN-82   1300      10
14 rows selected.
```

临时表(temporary table)

在数据库的实际应用中，我们常会遇到这样的问题，例如我们需要实现一个相当复杂的业务逻辑，这根本无法使用一条 select 语句来搞定；而是需要使用多条 select，包含多个 insert、update、delete 操作才能实现。那么，这时就有一个问题，我们在实现该业务逻辑时，需要分步完成，每一步都会生成中间结果，这些中间结果应该如何存储？我们可以将其存储在普通表中，但如果数据库有这样一种机制，也就是我们用完这些中间结果后，Oracle 会自动清除这些数据，岂不更好？数据库可基于会话或事务来保存这些中间结果。存储这些中间结果的对象称为“临时表”。

我们分别创建基于会话事务的临时表，先创建基于事务的临时表：

```
U_TEST1@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create global temporary table temp_emp on commit delete rows as
select * from emp;
Table created.
```

此时查看该临时表的数据，发现为空：

```
SCOTT@ora11g> select count(*) from temp_emp;
  COUNT(*)
----------
  0
```

我们将 emp 表中的数据插入该临时表中，并更新一条数据：

```
SCOTT@ora11g> insert into temp_emp select * from emp;
14 rows created.
SCOTT@ora11g> update temp_emp set empno = 1234 where empno = 7900;
1 row updated.
```

注意，此时该事务尚未结束，因此 temp_emp 中是存在数据的。

```
SCOTT@ora11g> commit;
Commit complete.
SCOTT@ora11g> select count(*) from temp_emp;
  COUNT(*)
----------
  0
```

一旦提交，当前事务即结束，Oracle 会自动清理 temp_emp 临时表中的数据，因此为空。

再来看创建基于会话的临时表：

```
SCOTT@ora11g> create global temporary table temp_emp_new on commit preserve rows
as select * from emp;
Table created.
```

由于该临时表是基于会话的，因此创建完毕就包含数据：

```
SCOTT@ora11g> select count(*) from temp_emp_new;
  COUNT(*)
----------
 14
```

然后我们执行一些 DML 操作并且提交：

```
SCOTT@ora11g> insert into temp_emp_new select * from emp;
14 rows created.
SCOTT@ora11g> update temp_emp_new set empno = 1234 where empno =7900;
2 rows updated.
SCOTT@ora11g> commit;
Commit complete.
SCOTT@ora11g> select count(*) from temp_emp_new;
  COUNT(*)
----------
 28
```

可见，temp_emp_new 临时表的数据依然存在。也就是说它跨越了多个事务，数据也没有被 Oracle 自动清除。接下来，我们切换一下用户，然后回来：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> conn scott/tiger;
Connected.
```

```
SCOTT@ora11g> select count(*) from temp_emp_new;
  COUNT(*)
----------
  0
```

此时，该表中就没有数据了。因为我们一旦切换用户，就开始了新会话，当前会话结束，临时表中的数据也就被数据库清空了。事务和会话的概念将在本章最后讲解。

数据库对象的删除

前面创建了表、索引、视图、序列、同义词以及临时表等数据库对象。这些对象的删除都使用 drop 命令。需要注意，一些对象依赖其他对象而创建。例如前面创建的索引、视图、同义词等都是基于测试表的。因此一旦测试表删除，这些对象将不复存在。如下：

```
SCOTT@ora11g> drop table tab_obj purge;
Table dropped.
SCOTT@ora11g> drop index ind_id;
drop index ind_id
         *
ERROR at line 1:
ORA-01418: specified index does not exist
```

3.7 本章涉及的相关概念

角色(role)

本章开头提到了 SQL 语言分类，其中的第三种就是 DCL 语言。DCL 是我们可以用来对数据库中对象以及数据库系统权限进行管理的语言。我们稍微深入一点，如果有这样的需求：我们有一批新招的开发工程师，他们每人都需要一个单独的数据库账户，但这些账户的权限又基本相同。那我们该怎样来简化这种权限分配操作呢？

角色其实就是权限的集合。为满足上面的需求，我们可将权限打包，然后授予各个数据库账户，这样显然就简单很多。本章前面用到的 resource 和 connect 其实就是最常用的两个角色。这两个角色包含的权限如下：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> select role,privilege from role_sys_privs where role in
('CONNECT','RESOURCE') order by role;
ROLE            PRIVILEGE
--------------  ----------------------------------------
CONNECT         CREATE SESSION
```

```
RESOURCE         CREATE CLUSTER
RESOURCE         CREATE INDEXTYPE
RESOURCE         CREATE OPERATOR
RESOURCE         CREATE PROCEDURE
RESOURCE         CREATE SEQUENCE
RESOURCE         CREATE TABLE
RESOURCE         CREATE TRIGGER
RESOURCE         CREATE TYPE
9 rows selected.
```

当然，也可以查阅 role_sys_privs 来了解数据库共提供了多少默认角色，以及这些角色各包含了哪些权限。需要注意，其中一个角色为 DBA，该角色包含大量的数据库管理权限和对数据库对象进行操作的权限，因此在生产系统中不要随意赋予一般数据库用户。另外，也可以自行创建角色并授予权限，如下：

```
SYS@ora11g> create role role_1;
Role created.
SYS@ora11g> grant connect,resource,create any index to role_1;
Grant succeeded.
SYS@ora11g> grant role_1 to scott;
Grant succeeded.
```

事务(transaction)

事务是数据库中的基本概念之一。简单来说，在数据库中，一个完整的操作称为一个事务。例如，我们想在银行账户 A 和 B 之间进行转账。那么，我们需要先在账户 A 中减去一定的金额，然后在账户 B 上进行相应的添加。显然，这两步操作需要全部完成，总金额才不会出现偏差。也就是说，在数据库中，一个事务要么全部完成，要么全部取消，没有中间状态。这种特性称为事务的四大特性之一——原子性(Atomicity)。其他三大特性为：一致性(Consistency)、隔离性(Isolation)和持久性(Durability)。

需要注意，DCL 语言就是用来控制事务的。commit 表示提交，也就是当前事务结束，所有操作都已完成，包括数据修改等。rollback 表示取消当前事务的所有操作，回退到事务开始前的样子。commit 和 rollback 是显式结束一个事务。DDL 和 DCL 则是隐式提交。也就是说，当 DDL 和 DCL 语句执行完毕，就表示事务结束，无须执行 commit 来显式提交；而 DML 操作则没有这样的情况。

这样，如果我们执行了大量 DML 操作，但此后没有执行 commit 或 rollback，又执行了一条 DDL 或 DCL，那么我们前面所有的 DML 操作也都将被提交。这一点一定要注意：对于 DML 操作，建议显式提交或回滚。

与事务相关的概念，还有一个称为隔离级别。按照国际标准，事务的隔离级别分为四级：未提交读(Read Uncommitted)、提交读(Read Committed)、重复读(Repeatable Read)以及串行

(Serializable)。Oracle数据库默认的隔离级别为提交读。也就是说，只有一个事务提交，其他事务才可以看到该事务的操作结果。如下：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> select * from tab_test;
 ID NAME
---------- ------------------------------
         5 test001
SCOTT@ora11g> update tab_test set id = 6;
1 row updated.
```

我们在当前窗口中执行 update 操作，但不提交，然后打开一个新窗口：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ sqlplus scott/tiger
SQL*Plus: Release 11.2.0.4.0 Production on Fri Apr 29 14:33:06 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SCOTT@ora11g> select * from tab_test;
 ID NAME
---------- ------------------------------
         5 test001
```

可见，此时我们查询到的还是 id = 5。然后我们回到第一个窗口，进行提交：

```
SCOTT@ora11g> commit;
Commit complete.
```

然后回到第二个窗口，重复刚才的 select 操作：

```
SCOTT@ora11g> /
 ID NAME
---------- ------------------------------
         6 test001
```

连接(process)与会话(session)

连接是用户进程和 Oracle 数据库实例之间建立的通信路径，可以是可用的进程间通信机制(在同一台机器上的用户进程和实例之间)，也可以是通过网络建立的。我们到目前为止通过 sqlplus 连接到数据库的方式，都是使用进程间通信机制建立的连接。数据库最多允许多少个连接由初始化参数 processes 来控制，默认为 150 个。

会话则表示当前用户登录数据库实例的状态。例如，前面使用scott用户登录数据库，系统就会为scott用户建立一个会话。会话从用户连接时开始，一直持续到用户断开连接或退出为止。数据库中最多允许的会话数目由初始化参数sessions控制。会话是基于连接而建立的。绝大多数情况下，会话和连接是一对一的关系，但有时也可以一对多。processes和sessions之间有一个简单的换算关系：

$$sessions = processes * 1.5 + 22$$

因此，sessions 默认值为 247。

当然，这些参数都可以根据系统实际的连接数目进行调整。

关于 Oracle 中的 SQL，除了本章所提到的基本内容之外，还有多表查询、子查询、集合操作、约束等内容。此外，还有高级 SQL 部分，例如分析函数、层次查询、正则表达式以及高级分组等。我们将在后续书籍中进行讲解。

SQL 是整个数据库管理的基本语言，无论是进行数据库开发，还是进行数据库管理，掌握基本的 SQL 语句都是最起码的要求。故建议各位初学者在测试环境中对本章的所有例子都进行实地测试运行，切实掌握基本 SQL 语句。

另外细心的读者可以注意到，在数据库中创建各种对象时，对象的命名是有一些规则的：例如，创建一个表，表名以 tab_开头；创建一个索引，索引名则以 ind_开头。其他对象也是如此。这样做的好处就是，我们可以直接从对象名中看出该对象是什么类型。如果再将该对象的用途和描述信息也放到对象名中，则会更好。但是数据库中对象的命名有如下限制：

对象名不能超过 30 个字符；

只能包含数字、字母、下划线、#和$；

不能以数字开头。

注意，关于本节的所有 SQL 例子，均可参考 Oracle 官方文档 *Database SQL Language Reference*。该文档包含了这些命令的语法结构以及相关例子。



第 4 章

Oracle 配置管理系列实验

完成数据库手工创建实验和 SQL 基础实验后，我们开始对数据库的配置进行调整设置。毕竟，手工方式创建的数据库，只是一个能够满足最基本需求的环境。要满足真正的生产环境的实际业务需求，还需要进行更多更深入的配置修改和设置。

4.1 控制文件多路复用实验

控制文件是 Oracle 数据库中最重要的文件之一。它记录了数据库的名称及其他关键配置，也记录了当前数据库中所有的数据文件和日志文件的位置及其状态等重要信息，是数据库启动过程中必须查找并且使用的关键文件。默认情况下，数据库中有两个控制文件，这两个控制文件内容相同，大小一致：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Tue May 3 09:04:45 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SYS@ora11g> startup
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size         2257840 bytes
Variable Size      520096848 bytes
Database Buffers   310378496 bytes
Redo Buffers       2371584 bytes
Database mounted.
Database opened.
SYS@ora11g> show parameter control_files;
NAME        TYPE  VALUE
------------------ --------- ------------------------------
control_files      string     /u01/oracle/oradata/ora11g/con
       trol01.ctl, /u01/oracle/fra/co
       ntrol2.ctl
```

可见，当前数据库中的两个控制文件是control01.ctl和control02.ctl，分别存放在/u01/oracle/oradata/ora11g目录和/u01/oracle/fra目录下。当然，也可以查看v$controlfile来获取当前数据库中控制文件的信息：

```
SYS@ora11g> select name from v$controlfile;
NAME
--------------------------------------------------------------------------
/u01/oracle/oradata/ora11g/control01.ctl
/u01/oracle/fra/control2.ctl
```

若想知道控制文件中存储了哪些内容，可执行如下查询：

```
SYS@ora11g> set pages 100;
```

```
SYS@ora11g> select distinct type from v$controlfile_record_section;
TYPE
----------------------------
FILENAME
TABLESPACE
RMAN CONFIGURATION
BACKUP CORRUPTION
PROXY COPY
FLASHBACK LOG
REMOVABLE RECOVERY FILES
DATAFILE
DELETED OBJECT
INSTANCE SPACE RESERVATION
RMAN STATUS
THREAD INSTANCE NAME MAPPING
GUARANTEED RESTORE POINT
ACM OPERATION
DATABASE
DATAFILE COPY
BACKUP DATAFILE
RECOVERY DESTINATION
DATAFILE HISTORY
COPY CORRUPTION
DATABASE INCARNATION
STANDBY DATABASE MATRIX
REDO LOG
TEMPORARY FILENAME
FOREIGN ARCHIVED LOG
CKPT PROGRESS
REDO THREAD
LOG HISTORY
OFFLINE RANGE
ARCHIVED LOG
BACKUP PIECE
MTTR
BACKUP SET
BACKUP REDOLOG
BACKUP SPFILE
RESTORE POINT
DATABASE BLOCK CORRUPTION
```

```
37 rows selected.
```

关于上述查询中用到的 v$controlfile 视图，可以查看官方文档 Database Reference 来了解更多信息。

接下来，我们来完成控制文件多路复用实验。所谓多路复用，实际上是将控制文件的个数从当前的两个改为三个，并将其存放在不同位置。这样，当其中一个或者两个控制文件出现问题或丢失时，数据库至少还保留有一个可用的控制文件。这样，恢复其他控制文件也就比较简单。但要注意，不管数据库有几个控制文件，数据库在启动时，会检查所有控制文件，这些控制文件的内容只有全部一致并且没有受损，数据库才能成功启动。

控制文件多路复用实验详细过程如下。

1. 生成 pfile 参数文件

关于 pfile 和 spfile，前面的 2.2 节已进行了说明。这里需要注意的是，这两个文件可以互相生成，本实验也借助了这一点。

```
SYS@ora11g> show parameter spfile;
NAME        TYPE        VALUE
----------- ----------- ------------------------------
spfile      string      /u01/oracle/product/11.2.0/dbs
                        /spfileora11g.ora
SYS@ora11g> create pfile from spfile;
File created.
```

上述第一条命令用来查看当前数据库是否使用了 spfile 这种二进制的参数文件。如果有输出结果，则表明使用了 spfile。然后利用 spfile 生成 pfile。默认情况下，pfile 和 spfile 都存放在 $ORACLE_HOME/dbs 目录下。pfile 以 init 开头，spfile 以 spfile 开头。

2. 关闭数据库

```
SYS@ora11g> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
```

实际上，数据库的关闭有多种选项。比如除了我们使用的 shutdown immediate 选项之外，还有 shutdown normal、shutdown transactional 以及 shutdown abort 等。在生产系统中，如果需要关闭数据库，我们建议使用 shutdown immediate 方式，这是最快捷也是最安全的关闭方式。

3. 修改 control_files 参数

```
SYS@ora11g> exit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
[oracle@rhel6 ~]$ cd $ORACLE_HOME/dbs
[oracle@rhel6 dbs]$ ls
hc_ora11g.dat  initora11g.ora   orapwora11g
init.ora       lkORA11G         spfileora11g.ora
[oracle@rhel6 dbs]$ vi initora11g.ora
```

这里的initora11g.ora就是我们前面用create pfile from spfile生成的pfile参数文件。spfileora11g.ora则是默认的spfile参数文件。pfile文件为文本格式，因此我们可直接修改。该文件内容如下(这里只需要修改control_files参数，因此忽略了pfile中的其他设置)：

```
*.control_files='/u01/oracle/oradata/ora11g/control01.ctl','/u01/oracle/fra
/control2.ctl'
```

我们需要在这一行后面再添加一份控制文件副本。修改后内容如下：

```
*.control_files='/u01/oracle/oradata/ora11g/control01.ctl','/u01/oracle/fra
/control2.ctl','/home/oracle/backup/control/control03.ctl'
```

上述内容中的阴影着重显示部分即为我们修改的内容。然后保存退出。也就是在当前 vi 的 insert 模式下，按 Esc 键退出 insert 模式，然后输入:x。注意修改的时候，不要忘了加逗号和引号。

4. 复制控制文件副本

第 3 步我们只是完成了参数修改，那实际上控制文件还是只有两份，因此我们需要复制一份。如下：

```
[oracle@rhel6 dbs]$ cp /u01/oracle/oradata/ora11g/control01.ctl
/home/oracle/backup/control/control03.ctl
```

5. 重新生成 spfile 并启动数据库

```
[oracle@rhel6 dbs]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Tue May 10 09:32:34 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SYS@ora11g> create spfile from pfile;
File created.
```

```
SYS@ora11g> startup;
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size          2257840 bytes
Variable Size       520096848 bytes
Database Buffers    310378496 bytes
Redo Buffers        2371584 bytes
Database mounted.
Database opened.
```

6. 查看实验修改结果

```
SYS@ora11g> show parameter control_files;
NAME        TYPE  VALUE
----------- ----------- ------------------------------
control_files       string /u01/oracle/oradata/ora11g/con
      trol01.ctl, /u01/oracle/fra/co
      ntrol2.ctl, /home/oracle/backu
      p/control/control03.ctl
```

至此，本实验完成。

需要注意，这里只是测试环境，如果是在生产环境中，建议控制文件的多个副本，分别存放在不同磁盘上。这样，一旦有一个或者两个磁盘受损，那至少还有一份可用的完好控制文件。控制文件最好三路复用。也就是说，数据库中应该有三份控制文件，并存放在不同磁盘上。

4.2 redo 日志组调整实验

Oracle数据库在每次进行数据修改操作时都会生成redo信息。简单来讲，redo相当于数据修改这样的操作的备份。有了redo的存在，使得我们在数据库出现故障需要进行数据恢复时，可使用redo来完成这样的工作。当然，redo如何在恢复过程中发挥作用，我们放在第 5 章中相应的实验来探讨。需要知道，redo日志在当前的数据库中默认有 3 组，每组两个redo日志文件，每个文件大小为 100M：

```
SYS@ora11g> select group#,sequence#,status,members,bytes/1024/1024 from
v$log;
    GROUP#  SEQUENCE#  STATUS            MEMBERS    BYTES/1024/1024
---------- ---------- ---------------- ---------- ---------------
         1         19  CURRENT                  2             100
         2         17  INACTIVE                 2             100
         3         18  INACTIVE                 2             100
```

这里的 group#表示组号，sequence#表示序列号。这里对 sequence 稍微解释一下，redo 日志组是一个循环使用的结构，也就是说，我们当前的数据库中有三个 redo 日志组，那么，数据库会在使用第 1 组时，将其 sequence 标记为 1，用完第 1 组之后，再用第 2 组，则第 2 组的 sequence 为 2，然后用第 3 组，则第 3 组的 sequence 为 3。当用完第 3 组后，数据库返回再用第 1 组，此时第 1 组的 sequence 为 4，依此类推。当前正在使用的日志组的状态为 current；刚刚用完还没有重新完成初始化以备下次使用的日志组，其状态为 active；已经完成初始化可供再次使用的日志组的状态为 inactive。members 列表示每个日志组中有几个日志文件。bytes 表示日志文件的大小，其单位为 byte，我们将其除两次 1024，则把单位转化为 MB。

细心的读者可以注意到，数据库中的redo日志组是循环复用的，那么，这样就会有一个问题。如果日志组被循环复用了一次之后，原来redo日志中的信息显然都已经被后来数据修改生成的redo信息覆盖掉了，那如果我们还想使用原来的redo信息，显然就没有办法了。redo信息对数据库的恢复极为重要，因此，我们应该在每个redo日志组用完之后对其进行备份才行。在我们当前的数据库中，默认是没有开启redo这种备份的。这种开启操作称为“启用数据库归档”，第 5 章会讲述如何开启redo备份。

查看当前数据库中 redo 日志的存放位置：

```
SYS@ora11g> select member from v$logfile;
MEMBER
---------------------------------------------------------------------------
-----
/u01/oracle/oradata/ora11g/redo01a.log
/u01/oracle/oradata/ora11g/redo01b.log
/u01/oracle/oradata/ora11g/redo02a.log
/u01/oracle/oradata/ora11g/redo02b.log
/u01/oracle/oradata/ora11g/redo03a.log
/u01/oracle/oradata/ora11g/redo03b.log

6 rows selected.
```

根据redo的生成机制，我们可以知道，数据库越繁忙，业务量越大，修改的数据量越多，redo生成的速度也越快，那么，当前数据库的redo日志组为 3 组，每组 2 个日志文件，每个文件大小为 100M，这样的设置能满足系统需求吗？

在 Oracle 数据库中，一旦数据发生修改操作，就一定会生成 redo。那么，如果 redo 组数太少，或者 redo 文件太小，导致 redo 日志组切换太快，就可能会影响数据库的正常运行。因此，我们需要确定当前的 redo 日志设置是否合理。

redo日志组设置是否合理的一个重要指标就是redo日志组的切换时间间隔。也就是一个redo日志组平均可以使用多长时间，然后才切换到下一个redo日志组。推荐的时间间隔为 10~15 分钟。建议在生产系统中，可以参考此推荐设置。那么，可以去哪里查看数据库redo日志组的

切换时间间隔呢？

可以查看 v$log_history 视图：

```
SYS@ora11g> select to_char(FIRST_TIME,'yyyy-mm-dd hh24:mi:ss') from
v$log_history;
TO_CHAR(FIRST_TIME,
-------------------
2016-04-22 11:11:21
2016-04-22 11:13:58
2016-04-22 11:15:01
2016-04-22 11:15:46
2016-04-22 11:17:26
2016-04-22 11:20:08
2016-04-22 11:21:26
2016-04-22 14:09:30
2016-04-27 10:17:53
2016-04-28 14:07:23
2016-04-28 16:53:06
2016-04-28 16:54:26
2016-04-28 16:54:34
2016-04-28 16:54:38
2016-04-28 16:56:29
2016-04-28 16:56:33
2016-04-29 09:10:38
2016-04-29 14:08:40

18 rows selected.
```

该视图中记录了每次日志切换的时间，可以根据任意上下两行的时间间隔来判断数据库的繁忙程度，以及redo日志组能否满足业务需求。因为这里为测试环境，因此数据库是相当空闲的)。我们尝试进行几次redo日志组的手工切换：

```
SYS@ora11g> alter system switch logfile;
System altered.
SYS@ora11g> /
System altered.
SYS@ora11g> /
System altered.
```

然后查看 v$log_history：

```
SYS@ora11g> select to_char(FIRST_TIME,'yyyy-mm-dd hh24:mi:ss') from
```

```
v$log_history;
    TO_CHAR(FIRST_TIME,
    -------------------
    2016-04-22 11:11:21
    2016-04-22 11:13:58
    2016-04-22 11:15:01
    2016-04-22 11:15:46
    2016-04-22 11:17:26
    2016-04-22 11:20:08
    2016-04-22 11:21:26
    2016-04-22 14:09:30
    2016-04-27 10:17:53
    2016-04-28 14:07:23
    2016-04-28 16:53:06
    2016-04-28 16:54:26
    2016-04-28 16:54:34
    2016-04-28 16:54:38
    2016-04-28 16:56:29
    2016-04-28 16:56:33
    2016-04-29 09:10:38
    2016-04-29 14:08:40
    2016-05-03 09:04:54
    2016-05-03 10:09:46
    2016-05-03 10:09:47

    21 rows selected.
```

可见，此时redo日志组的切换时间间隔显然太短了。当然，这里是手工触发，在实际生产环境中，如果redo日志组设置不合理，其切换时间间隔在 1 分钟之内也是有可能的。此时，在数据库的告警日志中，也会有相应的记录(这里需要打开一个新会话窗口)：

```
    [oracle@rhel6 control]$ tail -f
/u01/oracle/diag/rdbms/ora11g/ora11g/trace/alert_ora11g.log
    Thread 1 advanced to log sequence 21 (LGWR switch)
      Current log# 3 seq# 21 mem# 0: /u01/oracle/oradata/ora11g/redo03a.log
      Current log# 3 seq# 21 mem# 1: /u01/oracle/oradata/ora11g/redo03b.log
    Thread 1 cannot allocate new log, sequence 22
    Checkpoint not complete
      Current log# 3 seq# 21 mem# 0: /u01/oracle/oradata/ora11g/redo03a.log
      Current log# 3 seq# 21 mem# 1: /u01/oracle/oradata/ora11g/redo03b.log
    Thread 1 advanced to log sequence 22 (LGWR switch)
```

```
Current log# 1 seq# 22 mem# 0: /u01/oracle/oradata/ora11g/redo01a.log
Current log# 1 seq# 22 mem# 1: /u01/oracle/oradata/ora11g/redo01b.log
```

日志切换太快，会在告警日志文件中记录 checkpoint not complete 等信息。这里的 checkpoint 指检查点，Oracle 在每次切换日志时，都会执行一次检查点操作来完成 redo 当中记录的数据修改信息和数据文件中的信息的同步。切换太快，会导致这个同步操作无法及时完成。在告警日志中出现这样的问题，往往意味着 redo 日志组的设置不太合理。

接下来调整 redo 日志组，我们将其调整为 3 组，每组两个日志文件，每个日志文件大小为 500M。具体步骤如下。

1. 添加新的 redo 日志组

当前redo日志组的组数、大小、位置等信息在前面已经查询完毕，接下来添加新的redo日志组：

```
    SYS@ora11g> alter database add logfile group 4
('/u01/oracle/oradata/ora11g/redo04a.log','/u01/oracle/fra/redo04b.log') size
500M;
    Database altered.
    SYS@ora11g> alter database add logfile group 5
('/u01/oracle/oradata/ora11g/redo05a.log','/u01/oracle/fra/redo05b.log') size
500M;
    Database altered.
    SYS@ora11g> alter database add logfile group 6
('/u01/oracle/oradata/ora11g/redo056a.log','/u01/oracle/fra/redo06b.log') size
500M;
    Database altered.
```

数据库当前已经有了 1、2、3 几个组，因此我们添加 4、5、6 三组。另外需要注意，建议每个 redo 日志组中有两个 redo 文件，并且每个 redo 文件放在不同的磁盘上，这样可以提供 redo 日志的冗余，使其可靠性更高。然后查看当前 redo 日志组的设置：

```
    SYS@ora11g> select group#,sequence#,status,members,bytes/1024/1024 from
v$log;
        GROUP#  SEQUENCE#  STATUS            MEMBERS    BYTES/1024/1024
    ---------- ---------- ---------------- ---------- ---------------
             1         22  CURRENT                  2             100
             2         20  INACTIVE                 2             100
             3         21  INACTIVE                 2             100
             4          0  UNUSED                   2             500
             5          0  UNUSED                   2             500
```

```
         6          0  UNUSED                       2              500

6 rows selected.
```

可见，新添加的三组的状态均为 unused。

2. 删除旧的 redo 日志组

新日志组添加完成，但第 1 组 redo 日志的状态显示为 current，表明它是数据库当前正在使用的 redo 日志组。我们不能删除这样的 redo 日志组，为安全起见，应该删除状态为 inactive 的日志组。为此，需要先做一些日志切换：

```
SYS@ora11g> alter system switch logfile;
System altered.
SYS@ora11g> /
System altered.
SYS@ora11g> /
System altered.
```

再次查看 redo 日志组的状态：

```
SYS@ora11g> select group#,sequence#,status,members,bytes/1024/1024 from
v$log;
    GROUP#  SEQUENCE#  STATUS            MEMBERS     BYTES/1024/1024
---------- ---------- ---------------- ---------- ---------------
         1         22  INACTIVE                  2             100
         2         20  INACTIVE                  2             100
         3         21  INACTIVE                  2             100
         4         23  INACTIVE                  2             500
         5         24  INACTIVE                  2             500
         6         25  CURRENT                   2             500

6 rows selected.
```

此时，数据库当前使用的 redo 日志组为第 6 组(注意，如果读者在操作时不是第 6 组的话，可再做几次切换，从而保证与书中内容一致)，并且 1、2、3 组都处于 inactive 状态，我们可以执行删除操作：

```
SYS@ora11g> alter database drop logfile group 1;
Database altered.
SYS@ora11g> alter database drop logfile group 2;
Database altered.
SYS@ora11g> alter database drop logfile group 3;
```

```
Database altered.
```

此时，redo 日志组就只剩下新添加的三组了：

```
SYS@ora11g> select group#,sequence#,status,members,bytes/1024/1024 from
v$log;
    GROUP#  SEQUENCE#  STATUS            MEMBERS     BYTES/1024/1024
----------  ---------- ----------------  ----------  ---------------
         4          23 INACTIVE                   2              500
         5          24 INACTIVE                   2              500
         6          25 CURRENT                    2              500
```

需要注意，我们只是在数据库中删除了 1、2、3 这几组日志，而实际上其在操作系统上对应的日志文件并未删除，我们需要进行手工删除：

```
[oracle@rhel6 fra]$ cd /u01/oracle/oradata/ora11g/
[oracle@rhel6 ora11g]$ ls
control01.ctl  redo02a.log  redo03b.log   redo05a.log  temp01.dbf
redo01a.log    redo02b.log  redo04a.log   sysaux01.dbf  undotbs01.dbf
redo01b.log    redo03a.log  redo056a.log  system01.dbf  users01.dbf
[oracle@rhel6 ora11g]$ ls -lrt|grep redo
-rw-r----- 1 oracle oinstall 104858112 May  3 10:09 redo02a.log
-rw-r----- 1 oracle oinstall 104858112 May  3 10:09 redo02b.log
-rw-r----- 1 oracle oinstall 104858112 May  3 10:09 redo03b.log
-rw-r----- 1 oracle oinstall 104858112 May  3 10:09 redo03a.log
-rw-r----- 1 oracle oinstall 104858112 May  3 10:33 redo01a.log
-rw-r----- 1 oracle oinstall 104858112 May  3 10:33 redo01b.log
-rw-r----- 1 oracle oinstall 524288512 May  3 10:34 redo04a.log
-rw-r----- 1 oracle oinstall 524288512 May  3 10:34 redo05a.log
-rw-r----- 1 oracle oinstall 524288512 May  3 10:38 redo056a.log
[oracle@rhel6 ora11g]$ rm redo0[1-3]*
[oracle@rhel6 ora11g]$ ls -lrt|grep redo
-rw-r----- 1 oracle oinstall 524288512 May  3 10:34 redo04a.log
-rw-r----- 1 oracle oinstall 524288512 May  3 10:34 redo05a.log
-rw-r----- 1 oracle oinstall 524288512 May  3 10:38 redo056a.log
```

这里发现有一个小问题。我们在添加第 6 组 redo 日志时，两个日志文件名称应该分别为 redo06a.log 和 redo06b.log。我们将第一个文件的名称错写为 redo056a.log，因此对其进行重命名操作：

```
SYS@ora11g> alter database rename file
'/u01/oracle/oradata/ora11g/redo056a.log' to
```

```
'/u01/oracle/oradata/ora11g/redo06a.log';
    alter database rename file '/u01/oracle/oradata/ora11g/redo056a.log' to
'/u01/oracle/oradata/ora11g/redo06a.log'
    *
    ERROR at line 1:
    ORA-01511: error in renaming log/data files
    ORA-01621: cannot rename member of current log if database is open
    ORA-00312: online log 6 thread 1: '/u01/oracle/oradata/ora11g/redo056a.log'
    ORA-00312: online log 6 thread 1: '/u01/oracle/fra/redo06b.log'
```

这里的报错信息显示，当前数据库处于 open 状态，我们不能对文件进行重命名。既然如此，我们重启数据库到 mount 状态，再做修改(数据库的几个启动阶段将在本章最后进行详细说明)：

```
    SYS@ora11g> shutdown immediate;
    Database closed.
    Database dismounted.
    ORACLE instance shut down.
    SYS@ora11g> startup mount;
    ORACLE instance started.
    Total System Global Area  835104768 bytes
    Fixed Size      2257840 bytes
    Variable Size    520096848 bytes
    Database Buffers  310378496 bytes
    Redo Buffers      2371584 bytes
    Database mounted.
    SYS@ora11g> alter database rename file
'/u01/oracle/oradata/ora11g/redo056a.log' to
'/u01/oracle/oradata/ora11g/redo06a.log';
    alter database rename file '/u01/oracle/oradata/ora11g/redo056a.log' to
'/u01/oracle/oradata/ora11g/redo06a.log'
    *
    ERROR at line 1:
    ORA-01511: error in renaming log/data files
    ORA-01512: error renaming log file /u01/oracle/oradata/ora11g/redo056a.log -
    new file /u01/oracle/oradata/ora11g/redo06a.log not found
    ORA-27037: unable to obtain file status
    Linux-x86_64 Error: 2: No such file or directory
    Additional information: 3
```

这里又报错了，指出 redo06a.log 文件不存在，我们需要在操作系统上将原来的 redo056a.log

文件进行重命名，我们打开一个新窗口：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ cd /u01/oracle/oradata/ora11g/
[oracle@rhel6 ora11g]$ mv redo056a.log redo06a.log
```

然后回到 sqlplus 窗口，执行下列操作：

```
SYS@ora11g> alter database rename file
'/u01/oracle/oradata/ora11g/redo056a.log' to
'/u01/oracle/oradata/ora11g/redo06a.log';
Database altered.
SYS@ora11g> alter database open;
Database altered.
SYS@ora11g> select member from v$logfile;
MEMBER
--------------------------------------------------------------------------
/u01/oracle/oradata/ora11g/redo04a.log
/u01/oracle/fra/redo04b.log
/u01/oracle/oradata/ora11g/redo05a.log
/u01/oracle/fra/redo05b.log
/u01/oracle/oradata/ora11g/redo06a.log
/u01/oracle/fra/redo06b.log

6 rows selected.
```

此时，redo 日志组的文件名称就保持一致了。

这里还有一个细节，前面的实验没有触发。我们删除redo日志组时，建议删除状态为inactive的日志组。那么，如果想删除的redo日志组正好处于active状态，该如何处理？此时，可以执行如下操作：

```
SYS@ora11g> alter system checkpoint;
System altered.
```

这样，就可将处于 active 状态的 redo 日志组调整为 inactive。这条命令实际上是手工执行全局检查点事件，将所有仍在内存中的已修改(但仍未写到数据文件中)的数据都刷新到磁盘上。

4.3 ora-01555 重现实验

01555 错误是 Oracle 数据库中非常经典的错误之一，也是我们需要了解其发生原因和处理方式的错误之一。我们先来查看该错误的信息：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ oerr ora 01555
01555, 00000, "snapshot too old: rollback segment number %s with name \"%s\"
too small"
// *Cause: rollback records needed by a reader for consistent read are
// overwritten by other writers
// *Action: If in Automatic Undo Management mode, increase undo_retention
// setting. Otherwise, use larger rollback segments
```

oerr 是我们可在操作系统上使用的一条命令。我们可以用它来查看遇到的错误的相关信息。当然我们查看官方文档中的 Database Error Messages，也可以找到同样的错误信息。

01555 错误是与undo相关的一个错误。前面提到redo，说redo是数据修改操作的一个备份，与之相对应的，Oracle会在对数据进行修改之前，先保存一份未修改的数据，这称为undo。undo信息存放在undo表空间中。关于表空间的相关信息， 4.7 一节将进行说明。

Oracle 提供 undo 的目的很多，其中之一是保证实现一致性查询。前面的 3.7 一节中提到数据库一个很重要的名称：事务。事务在开始的时候，在修改数据之前，会首先去 undo 表空间中申请空间，将要修改的数据先存一份到 undo 表空间中，这些信息称为 undo 信息。这样，比如当我们正在修改一张表中的数据时，如果正好有其他会话或者事务来查看该表中的数据，就可以去 undo 中去查看修改之前数据的样子。毕竟，按照事务隔离性的要求，我们当前修改数据的事务还没有结束，其他会话或者事务看不到我们修改后的数据的样子，而只能查看数据在我们修改之前的样子，也就是“前镜像”。

如果 undo 中存在其他事务想查看的“前镜像”，则这些事务的查询就没有问题，这些查询称为一致性查询。但 undo 表空间中的空间也是循环使用的，只是 undo 信息会有一定的保留期限。那么问题来了，如果某一个事务想要查看的 undo 信息在 undo 表空间中不存在，会出现什么样的情况？本实验就是用来模拟此类问题的。

先查看当前数据库关于 undo 的相关信息：

```
SYS@ora11g> show parameter undo;
NAME                 TYPE         VALUE
------------------ ----------- ------------------------------
undo_management    string       AUTO
undo_retention     integer      900
undo_tablespace    string       UNDOTBS1
```

数据库中，与undo相关的初始化参数就上面这三个。当然，Oracle数据库中还有一类参数，称为隐含参数，这些参数是以下划线开头。我们用上面的show parameter无法查看隐含参数。隐含参数中也有关于undo的，作为一本针对零基础的读者的书，这里暂不对隐含参数做太多说明。

上面涉及三个隐含参数。undo_management表明undo管理默认使用的是自动管理模式，也

就是由Oracle自行管理，绝大部分情况下，不需要DBA进行干涉。undo_retention表示一个事务结束后，该事务生成的undo信息在undo表空间中保留的时间，默认为 900 秒。当然，这个时间Oracle在默认情况下是不予保证的。也就是说，实际情况下，undo信息在undo表空间中的保留时间可以是 900 秒，也可以大于或者小于它。如果你想强制保证undo信息的保留时间，需要执行如下操作：

```
SYS@ora11g> alter tablespace undotbs1 retention guarantee;
Tablespace altered.
```

这样，每个事务结束后，其生成的 undo 信息都会在 undo 表空间中至少保留 900 秒，然后才会被其他事务生成的 undo 信息覆盖。如果当前 undo 表空间中没有剩余空间，并且所有已经存储的 undo 信息都还没有保留够 900 秒，则数据库中所有新的事务都将失败，因为无法申请 undo 空间。因此，如果想保证 undo 的保留时间，undo 表空间就不能太小。

第三个参数 undo_tablespace 表示当前数据库正在使用的 undo 表空间。数据库中可以有多个 undo 表空间，但数据库当前使用的只能是一个。

接下来创建实验，模拟 01555 错误。

先创建一个新的 undo 表空间：

```
SYS@ora11g> create undo tablespace undo2 datafile
'/u01/oracle/oradata/ora11g/undo2.dbf' size 5M;
Tablespace created.
SYS@ora11g> alter system set undo_tablespace=undo2;
System altered.
```

并将该表空间设置为数据库当前的 undo 表空间。

再创建一个测试表：

```
SYS@ora11g> create table ora_01555 as select * from dba_objects;
Table created.
```

接下来声明一个游标，然后打开，但不查询游标中的数据：

```
SYS@ora11g> var x refcursor;
SYS@ora11g> exec open :x for select * from ora_01555;
PL/SQL procedure successfully completed.
```

然后，我们再打开一个会话，修改 ora_01555 中的数据：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Tue May 3 14:55:12 2016
Copyright (c) 1982, 2013, Oracle. All rights reserved.
```

```
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SYS@ora11g> begin
  2 for i in 1..10
  3 loop
  4 loop
  5 delete from ora_01555 where rownum <= 10000;
  6 exit when sql%rowcount = 0;
  7 commit;
  8 end loop;
  9 insert into ora_01555 select * from dba_objects;
 10 commit;
 11 end loop;
 12 end;
 13 /

PL/SQL procedure successfully completed.
```

注意，这里使用了一个简单的 PL/SQL 代码块来修改数据并提交。上述代码中的 2~13 为代码的行号，各位读者在操作时不要复制这些行号。

接下来回到第一个窗口中，打印我们前面声明的游标：

```
SYS@ora11g> print :x;
ERROR:
ORA-01555: snapshot too old: rollback segment number 12 with name
"_SYSSMU12_3592328248$" too small

no rows selected
```

显然，这个就是我们本节开头提到的 01555 错误。我们来看其详细含义：快照太旧，12 号回滚段太小。这个错误本身很简单，其实就是我们在前面使用 PL/SQL 代码块来修改数据的时候，由于 undo 表空间太小，导致我们此次修改直接将 undo 表空间中的空间都覆写了，因此在第一个窗口中打印游标中的数据时，就无法找到所需的 undo 信息，所以报错。

为消除这个错误，方法也比较简单，我们需要使用更大的 undo 表空间，然后重新执行游标的声明，并打印游标的数据即可。

```
SYS@ora11g> col FILE_ID for 99999;
SYS@ora11g> col file_name for a40
SYS@ora11g> select file_id,file_name,bytes/1024/1024 from dba_data_files;
FILE_ID FILE_NAME    BYTES/1024/1024
```

```
------- ---------------------------------------- --------
      5 /u01/oracle/oradata/ora11g/undo2.dbf           5
      1 /u01/oracle/oradata/ora11g/system01.dbf      325
      2 /u01/oracle/oradata/ora11g/sysaux01.dbf      325
      3 /u01/oracle/oradata/ora11g/undotbs01.dbf     200
      4 /u01/oracle/oradata/ora11g/users01.dbf       500
```

上述查询中，前两条命令用来控制select语句执行结果的显示格式。col file_id for 99999 表示file_id列为数值列，显示为五位有效数字；col file_name for a40 表示file_name为字符列，显示宽度为 40 个字符。查询结果中，编号为 3 的文件实际上就是原来的undotbs1，也就是初始的默认undo表空间。根据上述查询结果，我们可以知道该表空间中只有一个数据文件，大小为200MB。我们将当前数据库的undo表空间再切换回去：

```
SYS@ora11g> alter system set undo_tablespace=undotbs1;
System altered.
```

然后重新执行游标操作：

```
SYS@ora11g> var x refcursor;
SYS@ora11g>
SYS@ora11g> exec open :x for select * from ora_01555;
PL/SQL procedure successfully completed.
SYS@ora11g> print :x;
```

注意，这里输出结果太多，建议直接 Ctrl+C 键取消即可。最后删除前面创建的 undo2 表空间和测试表：

```
SYS@ora11g> drop tablespace undo2 including contents and datafiles;
Tablespace dropped.
SYS@ora11g> drop table ora_01555;
Table dropped.
```

关于 undo 表空间的更多细节，我们在本章的 4.7 节进行说明。

4.4 临时表空间组设置实验

临时表空间也是 Oracle 数据库的默认表空间之一。默认情况下，数据库中的临时表空间为：

```
SYS@ora11g> select tablespace_name from dba_tablespaces;
TABLESPACE_NAME
------------------------------
SYSTEM
```

```
SYSAUX
UNDOTBS1
TEMPTS1
USERS
```

这里的 TEMPTS1 就是默认的临时表空间。我们可通过如下查询确认这一结果：

```
SYS@ora11g> set line 100
SYS@ora11g> col PROPERTY_NAME for a25
SYS@ora11g> col PROPERTY_VALUE for a20
SYS@ora11g> col DESCRIPTION for a40
SYS@ora11g> select * from database_properties where PROPERTY_NAME like
'%TEMP%';
PROPERTY_NAME            PROPERTY_VALUE  DESCRIPTION
------------------------ --------------- ------------------------------------
DEFAULT_TEMP_TABLESPACE TEMPTS1          Name of default temporary tablespace
```

并且该表空间中只有一个临时文件：

```
SYS@ora11g> col file_name for a50
SYS@ora11g> select file_id,file_name from dba_temp_files where
tablespace_name='TEMPTS1';
   FILE_ID FILE_NAME
---------- --------------------------------------------------
         1 /u01/oracle/oradata/ora11g/temp01.dbf
```

但实际上，为提供更高的数据库性能，我们建议创建多个临时表空间，并组成临时表空间组供数据库使用。操作如下：

```
SYS@ora11g> create temporary tablespace temp001 tempfile
'/u01/oracle/oradata/ora11g/temp001.dbf' size 100m;
Tablespace created.
SYS@ora11g> create temporary tablespace temp002 tempfile
'/home/oracle/temp002.dbf' size 100m;
Tablespace created.
SYS@ora11g> alter tablespace temp001 tablespace group temp_grp;
Tablespace altered.
SYS@ora11g> alter tablespace temp002 tablespace group temp_grp;
Tablespace altered.
```

然后，将刚创建的 temp_grp 临时表空间组设置为数据库的默认临时表空间：

```
SYS@ora11g> alter database default temporary tablespace temp_grp;
```

```
Database altered.
SYS@ora11g> select * from database_properties where PROPERTY_NAME like
'%TEMP%';
PROPERTY_NAME            PROPERTY_VALUE  DESCRIPTION
------------------------ --------------- -----------------------------------
DEFAULT_TEMP_TABLESPACE  TEMP_GRP        Name of default temporary tablespace
```

建议，如果你所管理的生产系统非常繁忙，可参考上面的方式使用临时表空间组，并且每个临时表空间中可以有多个临时文件。为均衡磁盘 I/O，可将这些临时文件分散到不同磁盘上。

4.5 共享服务器连接模式配置实验

在当前测试环境，连接数据库时使用的都是 IPC 协议(也就是进程间通信)。实际上，就是在数据库所在的机器上，用户进程和数据库进程直接进行通信。但是，放到生产系统中，很多应用程序或者客户端往往都需要通过网络来连接到数据库。这时，我们就会用到监听程序(listener)。

用户进程通过监听程序与数据库实例建立连接，然后通过数据库的服务器进程来完成某些操作。这时，数据库有两种处理模式。第一种针对用户进程，每个用户进程都有专一的服务器进程来提供一对一服务。包括从磁盘上读取数据，并将查询结果返回给用户进程等。对于每个新的用户连接，都调用一个新的服务器进程为其提供服务。第二种就是先创建一批服务器进程，无论有多少个用户连接请求，都只使用这些服务器进程进行处理。前者称为专用服务器连接模式，后者称为共享服务器连接模式。Oracle 在默认情况下使用专用服务器连接模式。

显然，如果当前数据库处理的用户连接数量不大，并且每个用户连接往往又需要完成较多操作，此时使用专用服务器连接模式就较合适。但如果数据库应用拥有大量用户连接，但每个连接的操作又量级较轻的话，则共享服务器连接模式就更合适。

专用服务器连接模式为默认配置，无须设置，可以直接使用，但共享服务器连接模式需要专门配置。我们一步一步地讲述。

要完成该实验，需要完成如下步骤：

1. 查看当前监听的状态

前面在 1.5 节的最后创建了一个监听。该监听名为 listener，监听的服务器端口为 1521，该监听也称为默认监听。我们先了解一些管理监听的常用命令：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ lsnrctl status
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 03-MAY-2016 16:00:25
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
```

```
TNS-12541: TNS:no listener
 TNS-12560: TNS:protocol adapter error
  TNS-00511: No listener
   Linux Error: 111: Connection refused
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1521)))
TNS-12541: TNS:no listener
 TNS-12560: TNS:protocol adapter error
  TNS-00511: No listener
   Linux Error: 111: Connection refused
```

上述命令用来查看默认监听的状态，此时显示当前没有监听。实际上是因为我们没有启动监听，我们先启动监听：

```
[oracle@rhel6 ~]$ lsnrctl start;
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 03-MAY-2016 16:02:10
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Starting /u01/oracle/product/11.2.0/bin/tnslsnr: please wait...
TNSLSNR for Linux: Version 11.2.0.4.0 - Production
System parameter file is
/u01/oracle/product/11.2.0/network/admin/listener.ora
Log messages written to /u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date                03-MAY-2016 16:02:12
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/oracle/product/11.2.0/network/admin/listener.ora
Listener Log File
/u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
The listener supports no services
```

```
The command completed successfully
```

上述两条命令都没有指定监听的名称，因为我们操作的是默认监听。如果是非默认监听，则需要在命令后加上监听名。我们在第二步将会看到。

隔一分钟之后，再次查看监听状态：

```
[oracle@rhel6 ~]$ lsnrctl status
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 03-MAY-2016 16:03:13
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date                03-MAY-2016 16:02:12
Uptime                    0 days 0 hr. 1 min. 1 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/oracle/product/11.2.0/network/admin/listener.ora
Listener Log File
/u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
Services Summary...
Service "ora11g" has 1 instance(s).
  Instance "ora11g", status READY, has 1 handler(s) for this service...
The command completed successfully
```

Uptime表示监听运行的时间。一分钟后，可以看到执行上述命令之后有了Service的相关信息。可以看到，默认监听上有一个服务名为ora11g，该服务由实例ora11g提供，并且ora11g实例的状态为ready。实际上，用户连接数据库时，就是通过服务名来完成的。用户连接数据库时，需要提供要连接的服务名。而数据库一端会把自己提供的服务名注册到监听上(默认由Oracle数据库的PMON进程自动完成，该进程每隔一分钟工作一次)。这样，当监听程序收到用户连接请求时，可通过服务名将用户进程和提供该服务的数据库实例的服务器进程关联起来。

除了前面用到的 lsnrctl start、lsnrctl status 两个常用监听命令外，还有 reload、stop、services 等命令。比如，我们想查看有哪些服务名注册到默认监听上：

```
[oracle@rhel6 ~]$ lsnrctl services;
```

```
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 03-MAY-2016 16:13:38
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
Services Summary...
Service "ora11g" has 1 instance(s).
  Instance "ora11g", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
The command completed successfully
```

上面命令执行结果中的阴影着重显示部分表示，通过该监听连接到数据库上的连接都是专用服务器连接模式。

2. 创建第二个监听

接下来创建第二个监听，并使用共享服务器连接模式，通过该监听连接到数据库。

要创建监听，可以用手工方式，或者直接使用 netca/netmgr 图形化工具。下面列出两种操作方式，读者可任选其一，效果一样。

先看手工方式，实际上就是编辑 listener.ora 文件：

```
[oracle@rhel6 ~]$ vi $ORACLE_HOME/network/admin/listener.ora;
```

我们在该文件的最后，另起一行添加如下内容：

```
LSNR_2 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = rhel6)(PORT = 1526))
    )
  )
```

然后保存退出。

或者使用图形化方式：

```
[oracle@rhel6 ~]$ netca
Oracle Net Services Configuration:
No protocol specified
Error: null
Check the trace file for details:
/u01/oracle/cfgtoollogs/netca/trace_OraDb11g_home1-1605034PM2032.log
Oracle Net Services configuration failed.  The exit code is 1
```

这里报错，我们需要在 root 用户下执行 xhost +命令：

```
[oracle@rhel6 ~]$ exit
logout
[root@rhel6 Desktop]# xhost +
access control disabled, clients can connect from any host
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ netca
```

此时便启动图形化界面，如图 4-1 所示：



图 4-1 netca 页面

点击 Next，进入如图 4-2 所示的页面。

这里需要添加一个监听，因此保持默认，继续点击 Next(图 4-3)。



图 4-2 添加监听页面



图 4-3　监听命名页面

这里输入监听名 LSNR_2，点击 Next(图 4-4)：



图 4-4　连接协议选择页面

这里需要选择监听器所支持的连接协议，我们保持默认，点击 Next(图 4-5)。

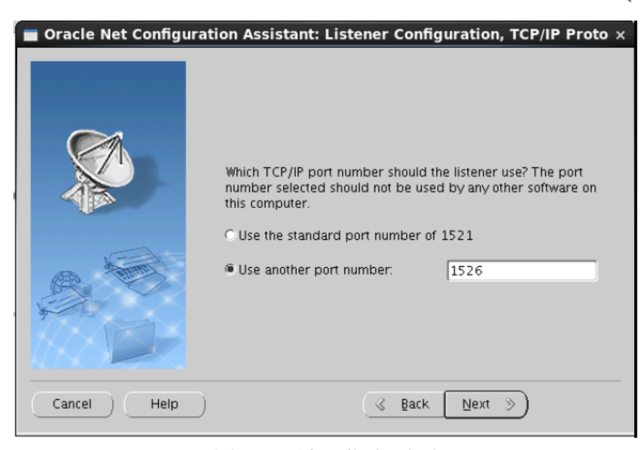


图 4-5　端口指定页面

这里需要我们确定 LSNR_2 这个监听所使用的端口。因为默认的 1521 端口已经被默认监听 listener 使用，因此需要指定新端口，这里指定为 1526。点击 Next(图 4-6)：

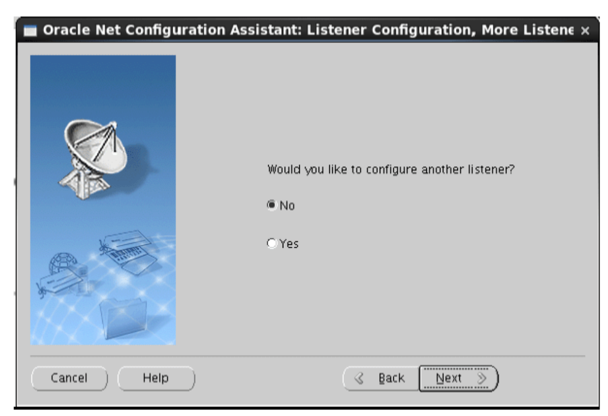


图 4-6 是否配置其他监听页面

系统询问是否还需要配置其他监听。这里不需要，保持默认，点击 Next(图 4-7)：

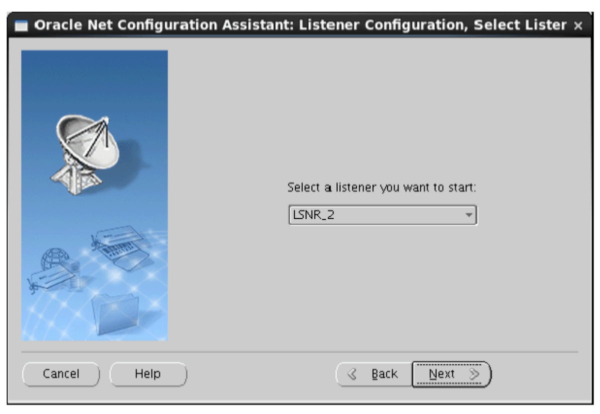


图 4-7 启动监听页面

选择一个监听启动，我们选择刚创建的 LSNR_2，点击 Next(图 4-8)：

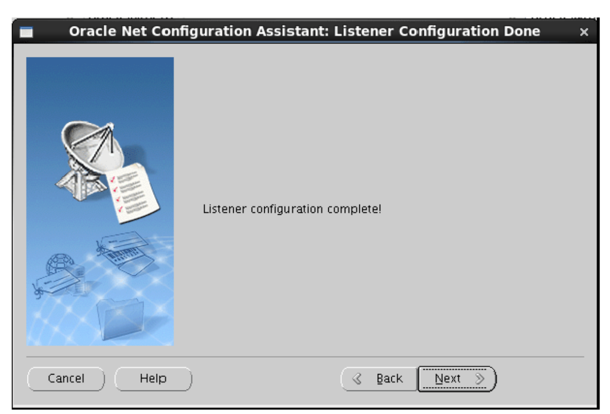


图 4-8 键盘配置完毕页面

监听器配置完毕，点击 Next(图 4-9)：



图 4-9　退出页面

点击 Finish，完成监听器配置。

此时，我们查看 listener.ora 文件，内容如下：

```
[oracle@rhel6 ~]$ cat $ORACLE_HOME/network/admin/listener.ora
# listener.ora Network Configuration File:
/u01/oracle/product/11.2.0/network/admin/listener.ora
# Generated by Oracle configuration tools.

LSNR_2 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = rhel6)(PORT = 1526))
    )
  )

ADR_BASE_LSNR_2 = /u01/oracle

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = rhel6)(PORT = 1521))
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
  )

ADR_BASE_LISTENER = /u01/oracle
```

可见，使用图形化工具创建监听与手工方式实际上是一样的，都是修改 listener.ora 文件。至于 netmgr 图形化方式，读者可以自己尝试使用。

3. 数据库服务的注册

第二个监听 LSNR_2 创建完毕后，我们来查看一下它的状态：

```
[oracle@rhel6 ~]$ lsnrctl status lsnr_2
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 03-MAY-2016 16:37:12
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1526)))
STATUS of the LISTENER
------------------------
Alias                    LSNR_2
Version                  TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date               03-MAY-2016 16:31:24
Uptime                   0 days 0 hr. 5 min. 48 sec
Trace Level              off
Security                 ON: Local OS Authentication
SNMP                     OFF
Listener Parameter File
/u01/oracle/product/11.2.0/network/admin/listener.ora
Listener Log File        /u01/oracle/diag/tnslsnr/rhel6/lsnr_2/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1526)))
The listener supports no services
The command completed successfully
```

数据库当前的服务注册在默认监听 listener 上，因此 LSNR_2 监听上是没有服务的。要使用这个监听来连接数据库，需将数据库服务注册在这个非默认监听上。我们需要执行如下操作：

```
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Tue May 3 16:38:51 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SYS@ora11g> show parameter local_listener;
NAME       TYPE        VALUE
---------- ----------- ------------------------------
local_     listener    string
SYS@ora11g> alter system set local_listener='LSNR_2';
```

```
alter system set local_listener='LSNR_2'
*
ERROR at line 1:
ORA-02097: parameter cannot be modified because specified value is invalid
ORA-00119: invalid specification for system parameter LOCAL_LISTENER
ORA-00132: syntax error or unresolved network name 'LSNR_2'
```

这里需要把数据库的初始化参数 local_listener 设置为我们刚创建的非模默认监听 LSNR_2。但报错了，我们要先执行如下操作，将 LSNR_2 的地址添加到 tns 文件中：

```
SYS@ora11g> exit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
[oracle@rhel6 ~]$ vi $ORACLE_HOME/network/admin/tnsnames.ora
 [oracle@rhel6 admin]$ vi $ORACLE_HOME/network/admin/tnsnames.ora
```

在该文件中添加如下内容：

```
LSNR_2 =
  (ADDRESS = (PROTOCOL = TCP)(HOST = rhel6)(PORT = 1526))
```

并保存退出，然后重新修改初始化参数 local_listener:

```
[oracle@rhel6 admin]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Tue May 3 16:42:13 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SYS@ora11g> alter system set local_listener='LSNR_2';
System altered.
```

然后查看监听 LSNR_2 的状态：

```
[oracle@rhel6 ~]$ lsnrctl status lsnr_2
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 03-MAY-2016 16:45:26
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1526)))
STATUS of the LISTENER
------------------------
Alias                     LSNR_2
Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
```

```
Start Date              03-MAY-2016 16:31:24
Uptime                  0 days 0 hr. 14 min. 1 sec
Trace Level             off
Security                ON: Local OS Authentication
SNMP                    OFF
Listener Parameter File
/u01/oracle/product/11.2.0/network/admin/listener.ora
Listener Log File       /u01/oracle/diag/tnslsnr/rhel6/lsnr_2/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1526)))
Services Summary...
Service "ora11g" has 1 instance(s).
  Instance "ora11g", status READY, has 1 handler(s) for this service...
The command completed successfully
```

可见，此时数据库的服务已经由 PMON 进程自动注册到 LSNR_2 上，同时，默认监听 listener 上则没有数据库服务了，如下：

```
[oracle@rhel6 ~]$ lsnrctl status
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 03-MAY-2016 16:46:57
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                   LISTENER
Version                 TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date              03-MAY-2016 16:02:12
Uptime                  0 days 0 hr. 44 min. 44 sec
Trace Level             off
Security                ON: Local OS Authentication
SNMP                    OFF
Listener Parameter File
/u01/oracle/product/11.2.0/network/admin/listener.ora
Listener Log File
/u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
The listener supports no services
The command completed successfully
```

4. 修改参数

创建完非默认监听后，接下来需要设置 Oracle 数据库的一些参数，从而启动共享服务器进程和分派进程：

```
SYS@ora11g> show parameter dispatcher;
NAME               TYPE       VALUE
------------------ ---------- ------------------------------
dispatchers        string
max_dispatchers    integer
```

与专用服务器连接模式不同，共享服务器连接模式除了共享服务器进程，还多了分派进程。分派进程用于接收用户请求，并将这些请求挂到请求队列中。此后，共享服务器进程来取出这些请求进行处理，并将响应挂到响应队列中，分派进程再将响应取出返回给用户进程。请求队列和响应队列都在内存中分配和使用空间。

上述参数中，dispatchers 表示当前数据库的分派进程个数；max_dispatchers 则表示最多可以有几个分派进程。

```
SYS@ora11g> show parameter shared_server;
NAME                       TYPE        VALUE
------------------------ ----------- ------------------------------
max_shared_servers         integer
shared_server_sessions     integer
shared_servers             integer     0
```

shared_servers 表示当前数据库中允许有几个共享服务器进程；max_shared_servers 表示最多允许有几个共享服务器进程；shared_server_sessions 则表示基于共享服务器进程的会话可以有多少个。

上述五个参数都需要我们进行设置。其中 shared_server_sessions 的数目包含在 sessions 参数设置的会话个数之内。因此，如果数据库启用共享服务器连接模式，则 sessions 参数可能也需要相应地进行调整。接下来修改这几个参数：

```
SYS@ora11g> create pfile from spfile;
File created.
SYS@ora11g> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SYS@ora11g> exit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
```

```
With the Partitioning, OLAP, Data Mining and Real Application Testing options
[oracle@rhel6 admin]$ vi $ORACLE_HOME/dbs/initora11g.ora
```

因为这里需要修改至少五个参数，因此我们在 pfile 中进行修改。在 initora11g.ora 文件末尾处添加如下内容：

```
dispatchers='(PROTOCOL=TCP)(DISPATCHERS=3)'
max_dispatchers=10
shared_servers=10
max_shared_servers=30
shared_server_sessions=100
```

这里，我们设置分派进程个数为三个，并且均为 tcp 连接协议，最大分派进程个数为 10 个。共享服务器进程个数为 10，最大为 30，基于共享服务器连接模式的会话个数最大为 100。

然后重新生成 spfile，并重启数据库：

```
[oracle@rhel6 admin]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Tue May 3 17:02:20 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SYS@ora11g> create spfile from pfile;
File created.
SYS@ora11g> startup;
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size      2257840 bytes
Variable Size    553651280 bytes
Database Buffers  276824064 bytes
Redo Buffers     2371584 bytes
Database mounted.
Database opened.
```

此时，在操作系统上，我们可以看到：

```
[oracle@rhel6 ~]$ ps -ef|grep ora_d0|grep -v grep
oracle    6119     1  0 17:03 ?        00:00:00 ora_d000_ora11g
oracle    6121     1  0 17:03 ?        00:00:00 ora_d001_ora11g
oracle    6123     1  0 17:03 ?        00:00:00 ora_d002_ora11g
[oracle@rhel6 ~]$ ps -ef|grep ora_s0|grep -v grep
oracle    6125     1  0 17:03 ?        00:00:00 ora_s000_ora11g
oracle    6127     1  0 17:03 ?        00:00:00 ora_s001_ora11g
oracle    6129     1  0 17:03 ?        00:00:00 ora_s002_ora11g
```

```
oracle    6131     1  0 17:03 ?        00:00:00 ora_s003_ora11g
oracle    6133     1  0 17:03 ?        00:00:00 ora_s004_ora11g
oracle    6135     1  0 17:03 ?        00:00:00 ora_s005_ora11g
oracle    6137     1  0 17:03 ?        00:00:00 ora_s006_ora11g
oracle    6139     1  0 17:03 ?        00:00:00 ora_s007_ora11g
oracle    6141     1  0 17:03 ?        00:00:00 ora_s008_ora11g
oracle    6143     1  0 17:03 ?        00:00:00 ora_s009_ora11g
```

其中，ora_d000_ora11g、ora_d001_ora11g、ora_d002_ora11g 为分派进程，后面的 ora_s000~009_ora11g 为共享服务器进程。前者为 3 个，后者为 10 个。

5. 测试

第二个监听和数据库参数已经调整完毕，接下来需要设置tns并进行测试了。先编辑tnsnames.ora文件：

```
[oracle@rhel6 ~]$ vi $ORACLE_HOME/network/admin/tnsnames.ora
```

在该文件中添加如下内容：

```
shared_test =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = rhel6 )(PORT = 1526))
    (CONNECT_DATA =
      (SERVER = shared )
      (SERVICE_NAME = ora11g )
    )
   )
```

注意，在执行上述操作时，上面的左括号需要缩进。

然后，我们使用 system 用户进行测试：

```
[oracle@rhel6 ~]$ sqlplus system/oracle@shared_test;
SQL*Plus: Release 11.2.0.4.0 Production on Tue May 3 17:11:44 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SYSTEM@shared_test> select server from v$session;
SERVER
---------
DEDICATED
DEDICATED
```

```
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
DEDICATED
SHARED

22 rows selected.
```

注意，上述命令结果中的阴影着重显示部分表示当前连接为共享服务器连接模式。查看此时 LSNR_2 上的服务：

```
[oracle@rhel6 ~]$ lsnrctl services lsnr_2;
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 03-MAY-2016 17:14:55
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1526)))
Services Summary...
Service "ora11g" has 1 instance(s).
  Instance "ora11g", status READY, has 4 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
      "D002" established:0 refused:0 current:0 max:1022 state:ready
         DISPATCHER <machine: rhel6.oracle.com, pid: 6123>
         (ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=48410))
      "D001" established:0 refused:0 current:0 max:1022 state:ready
         DISPATCHER <machine: rhel6.oracle.com, pid: 6121>
```

```
        (ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=41516))
"D000" established:1 refused:0 current:0 max:1022 state:ready
        DISPATCHER <machine: rhel6.oracle.com, pid: 6119>
        (ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=26203))
The command completed successfully
```

注意上述**阴影着重显示部分**中的数字 1，表示当前有一个连接是通过共享服务器连接模式连接到数据库实例上的。

4.6　表空间不足调整实验

在最初始状态下，Oracle 数据库中共有如下几个表空间：system、sysaux、undotbs1、tempts1、users。其中 users 表空间用来存储普通用户创建的对象。比如我们使用 scott 用户创建的表、索引等都放在 users 表空间中。但在生产系统中，一旦数据库承受的业务量增加，数据量也会随之增加，存储数据的表空间的剩余空间会越来越少。因此，作为一个 DBA，对数据库中各个表空间的利用情况及其剩余空间状况必须一直进行监控管理，并在某个表空间不足时进行调整。

我们先创建一个测试表空间，并在其中创建多个对象，然后了解当该表空间不足时，我们该如何处理。

首先创建测试表空间 tbs_test：

```
SYS@ora11g> create tablespace tbs_test datafile
'/u01/oracle/oradata/ora11g/tbs_test_01.dbf' size 100M;
Tablespace created.
```

然后创建测试表和索引：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create table tab_obj1 tablespace tbs_test as select * from
dba_objects;
Table created.
SCOTT@ora11g> create index ind_obj1 on tab_obj1(object_id) tablespace tbs_test;
Index created.
```

需要注意上述命令中的阴影着重显示部分，因为 scott 用户的默认表空间为 users：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> select username,DEFAULT_TABLESPACE from dba_users where
```

```
username='SCOTT';
    USERNAME       DEFAULT_TABLESPACE
    ------------- ------------------------------
    SCOTT          USERS
```

所以我们在 scott 用户下创建测试表和索引时，如果不指定 tablespace，则这些表和索引的数据都存储在 users 表空间中。

现在已经在表空间 tbs_test 中创建了表 tab_obj1 和索引 ind_obj1，那么 tbs_test 表空间还剩多少呢？这里要用到数据字典表 dba_free_space，该字典表记录了当前数据库中所有表空间的剩余空间情况：

```
    SYS@ora11g>
    select TABLESPACE_NAME,BYTES/1024/1024 from dba_free_space where
TABLESPACE_NAME='TBS_TEST';
    TABLESPACE_NAME   BYTES/1024/1024
    ----------------- ---------------
    TBS_TEST          96.6875
```

可知，tbs_test 表空间使用了不到 4M 的空间。我们接下来调整表中的数据量：

```
    SYS@ora11g> conn scott/tiger;
    Connected.
    SCOTT@ora11g> insert into tab_obj1 select * from tab_obj1;
    13529 rows created.
    SCOTT@ora11g> /
    27058 rows created.
    SCOTT@ora11g> /
    54116 rows created.
    SCOTT@ora11g> /
    108232 rows created.
    SCOTT@ora11g> /
    216464 rows created.
    SCOTT@ora11g> commit;
    Commit complete.
```

在 tab_obj1 中反复插入几次数据，然后查看 tbs_test 表空间的剩余空间情况：

```
    SCOTT@ora11g> conn / as sysdba
    Connected.
    SYS@ora11g>
    select TABLESPACE_NAME,BYTES/1024/1024 from dba_free_space where
TABLESPACE_NAME='TBS_TEST';
```

```
TABLESPACE_NAME      BYTES/1024/1024
-------------------- ---------------
TBS_TEST             44
```

如果再向 tab_obj1 中插入数据呢？

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> insert into tab_obj1 select * from tab_obj1;
insert into tab_obj1 select * from tab_obj1
*
ERROR at line 1:
ORA-01653: unable to extend table SCOTT.TAB_OBJ1 by 1024 in tablespace TBS_TEST
```

此处就报错了，显示已经无法在 tbs_test 表空间申请空间了。也就是说，该表空间中的剩余空间已经无法满足此次插入数据的存储需求了。

那么，如果在生产系统中遇到此类问题，该如何处理？

一般来说，大致有如下两种思路：

(1) 既然表空间剩余空间不足了，那么扩大该表空间就是了。

(2) 缩小该表空间中的现有对象或者移动到其他表空间，岂不也行？

通常来说，我们往往都是采用第一种思路来解决问题。但这样的思路在实际环境中有时行不通。例如，DBA 和存储管理人员不是同一个人，你作为 DBA，想要去扩展表空间，首先需要提申请，走流程，然后才可以扩展。结果你发现，这流程走下来，至少得两个星期。如果表空间扩展是十分急迫的事情，那这样的情况显然就不是第一种思路能应对的。因此第二种思路将更适合。

我们先分析第一种思路，现在，tbs_test 表空间剩余空间不足，我们可采用如下方式来扩大表空间：

1. 重置表空间中数据文件的大小

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> col FILE_NAME for a50
SYS@ora11g> select file_id,file_name from dba_data_files where
tablespace_name='TBS_TEST';
   FILE_ID FILE_NAME
---------- --------------------------------------------------
         5 /u01/oracle/oradata/ora11g/tbs_test_01.dbf
```

tbs_test 表空间只有这一个数据文件，并且文件编号为 5，我们调整该文件的大小：

```
SYS@ora11g> alter database datafile
'/u01/oracle/oradata/ora11g/tbs_test_01.dbf' resize 200M;
Database altered.
```

这样就调整了数据文件的大小。当然，也可以直接使用数据文件编号：

```
SYS@ora11g> alter database datafile 5 resize 200M;
Database altered.
```

然后查看 tbs_test 表空间的剩余空间情况：

```
SYS@ora11g>
select TABLESPACE_NAME,BYTES/1024/1024 from dba_free_space where
TABLESPACE_NAME='TBS_TEST';
TABLESPACE_NAME                BYTES/1024/1024
------------------------------ ---------------
TBS_TEST                       104
```

当然，resize 数据文件可以扩大，也可以缩小。

2. 打开表空间中现有数据文件的自动扩展

采用第一种方法，我们知道 tbs_test 表空间中只有一个数据文件，文件编号为 5。我们也可以打开该文件的自动扩展，这样，当使用完现有空间后，再申请空间时，Oracle 会自动扩展该表空间：

```
SYS@ora11g> alter database datafile 5 autoextend on;
Database altered.
```

那么，一旦打开数据文件的自动扩展，则扩展是否有上限？

```
SYS@ora11g> select FILE_ID,BYTES/1024/1024,AUTOEXTENSIBLE,MAXBYTES/1024/1024
from dba_data_files where file_id=5;
   FILE_ID BYTES/1024/1024 AUT MAXBYTES/1024/1024
---------- --------------- --- ------------------
         5             200 YES         32767.9844
```

这里的 autoextensible 列显示是否打开自动扩展，maxbytes 表示扩展到的最大值为多少，这里为 32GB。

如果该数据文件所在磁盘的剩余空间不多，任由数据文件扩展到 32GB 的话，则可能导致该磁盘的空间会被全面占用。因此，如果磁盘无法满足 32GB 的空间需求，建议在打开数据文件自动扩展的同时，设置其最大值：

```
SYS@ora11g> alter database datafile 5 autoextend on maxsize 10G;
```

```
Database altered.
SYS@ora11g> select FILE_ID,BYTES/1024/1024,AUTOEXTENSIBLE,MAXBYTES/1024/1024
from dba_data_files where file_id=5;
   FILE_ID  BYTES/1024/1024  AUT  MAXBYTES/1024/1024
----------  ---------------  ---  ------------------
         5              200  YES               10240
```

3. 添加数据文件

除了上面两种针对现有数据文件执行操作的方法外，还可以向该表空间中添加数据文件：

```
SYS@ora11g> alter tablespace tbs_test add datafile
'/u01/oracle/oradata/ora11g/tbs_test_02.dbf' size 100m;
Tablespace altered.
```

然后查看该表空间的剩余空间：

```
SYS@ora11g> select TABLESPACE_NAME,BYTES/1024/1024 from dba_free_space where
TABLESPACE_NAME='TBS_TEST';
TABLESPACE_NAME         BYTES/1024/1024
----------------------  ---------------
TBS_TEST                             99
TBS_TEST                            104
```

需要注意，dba_free_space 中的结果是按单个数据文件的剩余空间来显示的。而 tbs_test 表空间中现有两个数据文件，因此上述查询需要调整一下：

```
SYS@ora11g> select TABLESPACE_NAME,sum(BYTES)/1024/1024
from dba_free_space
where TABLESPACE_NAME='TBS_TEST'
group by TABLESPACE_NAME;
TABLESPACE_NAME        SUM(BYTES)/1024/1024
---------------------  --------------------
TBS_TEST                                203
```

我们先按照表空间进行分组，然后求和，这样就可以了。

接下来看一下第二种思路。

1. 将当前表空间中占用空间较大的对象移到其他表空间中

首先要确定 tbs_test 表空间中哪些对象占用的空间较大，并按大小进行排序：

```
SYS@ora11g> col SEGMENT_NAME for a20
SYS@ora11g> select segment_name,segment_type,bytes/1024/1024
```

```
from dba_segments
where tablespace_name='TBS_TEST'
order by bytes desc;
SEGMENT_NAME          SEGMENT_TYPE       BYTES/1024/1024
-------------------- ------------------ ---------------
TAB_OBJ1             TABLE                           80
IND_OBJ1             INDEX                           15
```

这里用到的数据字典表为 dba_segments，该字典表记录了当前数据库中所有占用空间的对象的大小。可见表 tab_obj1 占用了 80M 的空间，索引 ind_obj1 占用了 15M 的空间。如果我们把这些对象移动到其他表空间(例如 users 表空间)，则 tbs_test 表空间中的剩余空间不就多了？

```
SYS@ora11g> select TABLESPACE_NAME,BYTES/1024/1024 from dba_free_space where
TABLESPACE_NAME='USERS';
TABLESPACE_NAME       BYTES/1024/1024
--------------------- ---------------
USERS                         498.625
```

当前 users 表空间中剩余空间为接近 500M，因此，我们把表 tab_obj1 移到 users 表空间中：

```
SYS@ora11g> alter table scott.tab_obj1 move tablespace users;
Table altered.
```

这样，tbs_test 表空间剩余空间变多，users 表空间剩余空间变少。

但要注意，索引 ind_obj1 是基于表 tab_obj1 创建的。索引中存储了数据在表中的存储位置。表被移动了，那么索引的状态呢？

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> select index_name,status from ind where index_name='IND_OBJ1';
INDEX_NAME            STATUS
--------------------- --------
IND_OBJ1              UNUSABLE
```

也就是说，一旦表被移动，则基于该表的所有索引的状态都会变为 unusable，也就是不可用。因此，我们要重建索引：

```
SCOTT@ora11g> alter index ind_obj1 rebuild online;
Index altered.
SCOTT@ora11g> select index_name,status from ind where index_name='IND_OBJ1';
INDEX_NAME            STATUS
--------------------- --------
IND_OBJ1              VALID
```

当然，你也可以借重建索引的机会将其移动到 users 表空间中：

```
SCOTT@ora11g> alter index ind_obj1 rebuild online tablespace users;
Index altered.
```

2. 删除或者备份表空间中某些对象的历史数据

如果我们知道表空间中的有些对象包含了大量历史数据或者几乎用不到的数据，那么，在这些对象所在的表空间剩余空间不足时，我们可将这些数据删除，或者备份出来(以备将来其他人使用)。这样，也可以增加表空间的剩余空间。

例如，如果知道 tab_obj1 中有些数据已经不再会被用到了，可将这些数据删除：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> select owner,SEGMENT_NAME,SEGMENT_TYPE,BYTES/1024/1024
from dba_segments where segment_name in ('TAB_OBJ1','IND_OBJ1');
OWNER        SEGMENT_NAME   SEGMENT_TYPE    BYTES/1024/1024
------------ -------------- --------------- ---------------
SCOTT        IND_OBJ1       INDEX                         8
SCOTT        TAB_OBJ1       TABLE                        45
```

这是当前这两个对象占用的空间大小，表为 45M，索引为 8M。

```
SYS@ora11g> select count(*) from scott.tab_obj1;
  COUNT(*)
----------
    432928
SYS@ora11g> delete from scott.tab_obj1 where rownum < 420000;
419999 rows deleted.
SYS@ora11g> commit;
Commit complete.
SYS@ora11g> select owner,SEGMENT_NAME,SEGMENT_TYPE,BYTES/1024/1024 from
dba_segments where segment_name in ('TAB_OBJ1','IND_OBJ1');
OWNER        SEGMENT_NAME   SEGMENT_TYPE    BYTES/1024/1024
------------ -------------- --------------- ---------------
SCOTT        IND_OBJ1       INDEX                         8
SCOTT        TAB_OBJ1       TABLE                        45
```

有趣的是，我们删除了表中的绝大部分数据，并且提交，但表和索引占用的空间依然没有变化。也就是说，虽然数据删除了，但占用的空间却没有释放。那我们来释放一下。由于现在 tab_obj1 表在 users 表空间中，因此我们删除该表，重建测试表和索引，然后执行相应的释放空间的命令：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> drop table tab_obj1 purge;
Table dropped.
SCOTT@ora11g> create table tab_obj1 tablespace tbs_test as select * from dba_objects;
Table created.
SCOTT@ora11g> insert into tab_obj1 select * from tab_obj1;
13529 rows created.
SCOTT@ora11g> /
27058 rows created.
SCOTT@ora11g> /
54116 rows created.
SCOTT@ora11g> /
108232 rows created.
SCOTT@ora11g> /
216464 rows created.
SCOTT@ora11g> select count(*) from tab_obj1;
  COUNT(*)
----------
    432928
SCOTT@ora11g> commit;
Commit complete.
SCOTT@ora11g> create index ind_obj1 on tab_obj1(object_id) tablespace tbs_test;
Index created.
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> select owner,SEGMENT_NAME,SEGMENT_TYPE,BYTES/1024/1024 from dba_segments where segment_name in ('TAB_OBJ1','IND_OBJ1');
OWNER        SEGMENT_NAME     SEGMENT_TYPE      BYTES/1024/1024
------------ ---------------- ----------------- ---------------
SCOTT        TAB_OBJ1         TABLE                          45
SCOTT        IND_OBJ1         INDEX                           8
```

接下来开始删除数据：

```
SYS@ora11g> delete from scott.tab_obj1 where rownum < 420000;
419999 rows deleted.
SYS@ora11g> commit;
Commit complete.
```

然后开始释放空间：

```
SYS@ora11g> alter table scott.tab_obj1 shrink space;
alter table scott.tab_obj1 shrink space
*
ERROR at line 1:
ORA-10636: ROW MOVEMENT is not enabled
```

这里报错，因为我们释放空间的时候，需要移动表中的数据，从而使数据的存储更紧凑，所以需要启用行移动：

```
SYS@ora11g> alter table scott.tab_obj1 enable row movement;
Table altered.
SYS@ora11g> alter table scott.tab_obj1 shrink space;
Table altered.
```

然后查看表和索引的大小：

```
SYS@ora11g> select owner,SEGMENT_NAME,SEGMENT_TYPE,BYTES/1024/1024
from dba_segments where segment_name in ('TAB_OBJ1','IND_OBJ1');
OWNER        SEGMENT_NAME   SEGMENT_TYPE    BYTES/1024/1024
------------ -------------- --------------- ---------------
SCOTT        TAB_OBJ1       TABLE                    1.4375
SCOTT        IND_OBJ1       INDEX                         8
```

可见，表确实变小了，但是索引还没有发生变化，我们重建索引：

```
SYS@ora11g> alter index scott.ind_obj1 rebuild online;
Index altered.
SYS@ora11g> select owner,SEGMENT_NAME,SEGMENT_TYPE,BYTES/1024/1024
from dba_segments where segment_name in ('TAB_OBJ1','IND_OBJ1');
OWNER        SEGMENT_NAME    SEGMENT_TYPE     BYTES/1024/1024
------------ --------------- ---------------- ---------------
SCOTT        TAB_OBJ1        TABLE                     1.4375
SCOTT        IND_OBJ1        INDEX                      .3125
```

当然，也可在释放表空间的同时，也级联释放索引中的空间：

```
SYS@ora11g> alter table scott.tab_obj1 shrink space cascade;
Table altered.
```

此时，tbs_test 表空间中的剩余空间为：

```
SYS@ora11g> select TABLESPACE_NAME,sum(BYTES)/1024/1024
```

```
from dba_free_space
where TABLESPACE_NAME='TBS_TEST'
group by TABLESPACE_NAME;
TABLESPACE_NAME        SUM(BYTES)/1024/1024
--------------------- --------------------
TBS_TEST                             296.25
```

3. 删除某些对象

这种方式就比较简单了，假如你发现 tab_obj1 这个对象已经不会再用，那么直接把这个表删除即可。这样也可以释放空间。但在生产系统中，千万要注意，在删除对象或者删除数据时，一定要确保这些数据或者对象以后肯定不会再被用到。你需要找相应的业务人员来保证这一点。如果无法确认，则要考虑备份数据。例如，可使用 exp 或者 expdp 来备份表 tab_obj1 中的数据，然后删除表(这里以 exp 为例)：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ exp scott/tiger tables=tab_obj1 file=dump1.dmp;
Export: Release 11.2.0.4.0 - Production on Wed May 4 10:56:19 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit
Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Export done in US7ASCII character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)
About to export specified tables via Conventional Path ...
. . exporting table                   TAB_OBJ1     12929 rows exported
EXP-00091: Exporting questionable statistics.
Export terminated successfully with warnings.
```

上面执行结果中包含 EXP-00091 错误，实际上是因为操作系统字符集和当前数据库的字符集不一致造成的。我们稍加处理即可。

首先获取当前数据库的字符集：

```
SCOTT@ora11g> select userenv('language') from dual;
USERENV('LANGUAGE')
----------------------------------------------------
AMERICAN_AMERICA.AL32UTF8
```

然后设置操作系统的字符集：

```
[oracle@rhel6 ~]$ export NLS_LANG=AMERICAN_AMERICA.AL32UTF8
[oracle@rhel6 ~]$ exp scott/tiger tables=tab_obj1 file=dump1.dmp;
```

```
Export: Release 11.2.0.4.0 - Production on Wed May 4 11:00:35 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit
Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Export done in AL32UTF8 character set and AL16UTF16 NCHAR character set
About to export specified tables via Conventional Path ...
. . exporting table                    TAB_OBJ1     12929 rows exported
Export terminated successfully without warnings.
```

再删除表 tab_obj1：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> drop table tab_obj1 purge;
Table dropped.
```

当以后再需要用到该表时，导入数据即可：

```
[oracle@rhel6 ~]$ imp scott/tiger tables=tab_obj1 file=dump1.dmp;
Import: Release 11.2.0.4.0 - Production on Wed May 4 11:06:02 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit
Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Export file created by EXPORT:V11.02.00 via conventional path
import done in AL32UTF8 character set and AL16UTF16 NCHAR character set
. importing SCOTT's objects into SCOTT
. importing SCOTT's objects into SCOTT
. . importing table                  "TAB_OBJ1"     12929 rows imported
Import terminated successfully without warnings.
```

4.7　本章涉及的相关概念

表空间与数据文件

表空间(tablespace)是数据库中的一个逻辑对象，用来管理数据文件。Oracle 中的数据(包括系统管理用的数据和用户自己插入或者修改后的数据)都存储在数据文件中。一旦数据库中数据文件太多，则对每个数据文件进行单独管理就会显得麻烦。因此 Oracle 引入表空间的概念。一个表空间中可以有一个或多个数据文件。有且只能有一个数据文件的表空间称为大文件(bigfile)表空间，该文件最大可以到 32TB。我们前面创建的 tbs_test 为普通表空间，也称为小

文件(smallfile)表空间。默认情况下，创建的表空间都为普通表空间，这样的表空间中可以包含多个数据文件，每个数据文件最大可以到 32GB。

默认情况下，数据库有 system、sysaux、undotbs1、tempts1、users 五个表空间。system 为系统表空间，由数据库管理所用，主要用于存放数据字典表等信息，建议不要在该表空间中创建任何普通用户的对象。sysaux 称为辅助表空间，用来存储与性能相关的一些对象。undotbs1 为 undo 表空间，用于存储数据修改时生成的 undo 信息。tempts1 为临时表空间，主要用来存储临时表的数据；在内存中进行数据排序时，如果内存不够大，临时表空间也可以用来存储排序的中间结果。users 表空间为默认的用户表空间，用来存储用户的业务数据。

段、区、块

段(segment)，简单来讲，一个表就是一个段，一个索引也是一个段。也就是占用空间的对象称为段。当然，这里不讨论分区的情况。在 Oracle 数据库中，常见的段有四种：数据段(表)、索引段(索引)、临时段(临时表空间中的对象)以及 undo 段(undo 表空间中的对象)。

区(extent)，数据文件中一串连续的块。区是数据库分配空间时的基本单位。也就是说，我们创建一个表，会分配一个或者多个区。

块(block)，数据文件中数据存储和磁盘 I/O 的基本单位。默认大小为 8192 字节，也就是 8KB。相关的初始化参数为：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> show parameter db_block_size;
NAME              TYPE      VALUE
----------------- --------- -------------
db_block_size     integer            8192
```

为防止磁盘 I/O 浪费，数据库中 block 的大小基本都是操作系统 block 大小的整数倍。

告警日志文件

告警日志文件也是数据库中的重要文件之一。它自数据库启动开始，记录数据库启动的详细过程、数据库结构的变化、日志切换的记录、数据库 block 损坏的信息等。很多时候，当数据库遇到问题或者故障，DBA 第一反应往往就是去查看告警日志。在 Oracle 11g 中，该文件有两份，分别是文本格式和 XML 格式。我们通常查看的是文本格式。可用如下查询来确定告警日志文件所在的位置：

```
SYS@ora11g> col name for a10
SYS@ora11g> col value for a50
SYS@ora11g> select * from v$diag_info where name ='Diag Trace';
   INST_ID NAME       VALUE
---------- ---------- --------------------------------------------------
         1 Diag Trace /u01/oracle/diag/rdbms/ora11g/ora11g/trace
```

告警日志文件以 alert 开头。我们打开一个新窗口：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ cd /u01/oracle/diag/rdbms/ora11g/ora11g/trace
[oracle@rhel6 trace]$ ls -lrt |grep alert
-rw-r----- 1 oracle oinstall 97537 May  4 14:26 alert_ora11g.log
```

这里的 alert_ora11g.log 就是当前数据库的告警日志文件。

跟踪文件

每个服务器进程和 oracle 的后台进程都可以写入相关联的跟踪文件。当进程检测到内部错误时，进程会将有关该错误的信息都存储到相应的跟踪文件中。跟踪文件和告警日志文件在同一目录中。并且文件名中往往包含对应进程的名称。

数据库启动的三个阶段

前面的 2.2 一节中提到了实例的概念。数据库启动，实际上启动的就是实例。数据库完整启动的三个阶段为：

1. nomount 状态

这一步，实际上就是完成实例的启动。当然，我们知道实例包含后台进程和内存，因此，数据库从关闭状态到 nomount 状态，要完成如下工作：

- 查找参数文件，先查找 spifle，如果没有，查找 pfile；
- 启动后台进程；
- 分配内存；
- 打开告警日志文件和跟踪文件。

数据库从关闭状态到 nomount 状态，可以使用如下命令来完成：

```
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Wed May 4 14:45:17 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SYS@ora11g>startup nomount;
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size          2257840 bytes
Variable Size       553651280 bytes
Database Buffers    276824064 bytes
Redo Buffers        2371584 bytes
```

2. mount 状态

启动实例后，oracle 数据库会按参数文件中 control_files 参数的设置，找到所有控制文件，在确定所有控制文件都完好并且内容一致后，将控制文件的内容加载到内存，并从控制文件中获取所有数据文件和日志文件的名称及其位置，但不做一致性检查。

将数据库从 nomount 状态调整到 mount 状态，命令如下：

```
SYS@ora11g> alter database mount;
Database altered.
```

3. open 状态

接下来，Oracle 需要检查所有数据文件和日志文件的状态，如果这些文件的实际状态和控制文件中记录的一致，则打开数据库，允许普通用户连接数据库并执行各种操作。

将数据库从 mount 状态调整到 open 状态，命令为：

```
SYS@ora11g> alter database open;
Database altered.
```

注意，重启数据库可以使用 startup force 命令，它等价于 shutdown abort + startup。

游标(cursor)

在本章的 4.3 一节中创建了一个游标来查询数据。在 Oracle 中，游标实际上是一个查询的别名，我们可以先声明一个游标，然后打开，接下来一条一条地取出游标中的数据，操作完毕后再关闭游标。游标在 PL/SQL 中相当常用，我们在后续书籍中再详细描述。

4.8 本章用到的 Linux 命令

cp　复制命令
tail　查看文件的命令，相应的还有 head、more、less 等
mv　移动文件或者文件夹的命令
rm　删除文件或者文件夹的命令



第5章

备份恢复系列实验

在生产环境中，DBA 大多在业务系统上线后才介入，然后开始数据库运维工作。当数据库投入正式运营后，作为DBA，一定需要考虑当数据库出现问题或故障时，我们该如何处理。我们都知道墨菲定律：如果事情有变坏的可能，不管这种可能性有多小，它总会发生。因此对于 DBA 来说，永远不要奢望你管理的数据库肯定不出问题，而要未雨绸缪，在数据库还没有发生任何故障的时候，就考虑数据库可能出现什么样的问题，以及在出现问题之后该如何处理。这就是备份恢复要关注的内容。

前面完成了数据库的创建以及一些重要的设置，本章分析如何对数据库进行备份，以及在出现一些常见故障时，一般应该如何处理。

5.1 归档与闪回开启实验

4.2 一节中提到了 redo，并且对 redo 日志组做了相关的实验，你可以参考该实验来调整生产系统中 redo 日志组的设置。当时我们预留了一个问题，这里完成数据库归档启动。

Oracle 的另一个特性是支持闪回(flashback)。所谓闪回，指我们可通过一些简单设置，或者利用数据库本身的一些特性，将某个表甚至整个数据库回退到过去某一时间点的状态。具体的闪回内容和相关实验将在 5.8 一节中详细介绍。

某些时候，我们可以把归档启动之后生成的 redo 日志的备份(也称为归档日志)和实现数据库闪回所需的日志(称为闪回日志——flashback log)存放在同一目录下，因此这里将闪回和数据库归档两个操作合二为一。

先查看当前数据库是否开启闪回和归档：

```
SYS@ora11g> archive log list;
Database log mode              No Archive Mode
Automatic archival             Disabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     28
Current log sequence           30
```

这里显示 No Archive Mode，意味着当前数据库没有开启归档。也可以使用如下方式查看：

```
SYS@ora11g> select log_mode from v$database;
LOG_MODE
------------
NOARCHIVELOG
```

然后再查看当前数据库是否开启闪回功能：

```
SYS@ora11g> select flashback_on from v$database;
FLASHBACK_ON
------------------
NO
```

将上面两个查询合二为一，就可以用一条命令查看当前数据库是否开启闪回和归档：

```
SYS@ora11g> select log_mode,flashback_on from v$database;
LOG_MODE     FLASHBACK_ON
------------ ------------------
```

```
NOARCHIVELOG NO
```

可见，当前数据库无论是闪回还是归档，都没有开启。

要开启闪回或者归档，我们需要设置存放归档日志和闪回日志的位置以及空间。因此，先查看如下参数：

```
SYS@ora11g> show parameter db_recovery_file_dest;
NAME                         TYPE              VALUE
---------------------------- ----------------- ------------------------
db_recovery_file_dest        string            /u01/oracle/fra
db_recovery_file_dest_size   big integer 4G
```

第一个参数指示存放位置，第二个参数指示可以使用的空间大小，这两个参数在前面 2.1 一节中的第 5 步中已经设置完毕，这里直接使用即可。如果这两个参数没有设置，则需要先设置大小，然后设置位置。

接下来关闭数据库：

```
SYS@ora11g> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
```

然后将数据库重启到 mount 阶段：

```
SYS@ora11g> startup mount;
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size          2257840 bytes
Variable Size       553651280 bytes
Database Buffers    276824064 bytes
Redo Buffers        2371584 bytes
Database mounted.
```

然后开启闪回和归档，注意，这里有顺序要求：必须先开启归档，后开启闪回：

```
SYS@ora11g> alter database archivelog;
Database altered.
SYS@ora11g> alter database flashback on;
Database altered.
SYS@ora11g> alter database open;
Database altered.
SYS@ora11g>  select log_mode,flashback_on from v$database;
LOG_MODE     FLASHBACK_ON
```

```
------------ ------------------
ARCHIVELOG   YES
```

必须在数据库处于 mount 状态下完成数据库归档的开启，并且之前的数据库关闭必须为一致性关闭。数据库闪回则可以在数据库处于 mount 或者 open 状态下开启。并且必须在数据库归档开启之后。

默认情况下，开启数据库归档后，数据库会自动启动四个归档进程：

```
SYS@ora11g> show parameter log_archive_max_processes;
NAME                         TYPE        VALUE
---------------------------- ----------- ------
log_archive_max_processes    integer          4
```

我们在操作系统上可以看到这四个进程：

```
[oracle@rhel6 ~]$ ps -ef|grep ora_arc|grep -v grep
oracle    4609     1  0 15:27 ?        00:00:00 ora_arc0_ora11g
oracle    4611     1  0 15:27 ?        00:00:00 ora_arc1_ora11g
oracle    4613     1  0 15:27 ?        00:00:00 ora_arc2_ora11g
oracle    4615     1  0 15:27 ?        00:00:00 ora_arc3_ora11g
```

接下来手工切换几次日志，然后查看生成的归档日志：

```
SYS@ora11g> alter system switch logfile;
System altered.
SYS@ora11g> /
System altered.
SYS@ora11g> /
System altered.
SYS@ora11g> desc v$log;
 Name              Null?    Type
 ---------------- --------- ----------------------------
 GROUP#            NUMBER
 THREAD#           NUMBER
 SEQUENCE#         NUMBER
 BYTES             NUMBER
 BLOCKSIZE         NUMBER
 MEMBERS           NUMBER
 ARCHIVED          VARCHAR2(3)
 STATUS            VARCHAR2(16)
 FIRST_CHANGE#     NUMBER
 FIRST_TIME        DATE
```

```
 NEXT_CHANGE#     NUMBER
 NEXT_TIME        DATE
SYS@ora11g> select GROUP#,SEQUENCE#,ARCHIVED,STATUS from v$log;
   GROUP#  SEQUENCE#  ARC STATUS
---------- ---------- --- ----------------
        4          32  YES INACTIVE
        5          33  NO  CURRENT
        6          31  YES INACTIVE
```

可以看到，当前序列号为 31,32 的日志都已经完成归档(参见 archived 列)。我们去查看生成的归档日志：

```
[oracle@rhel6 ~]$ cd /u01/oracle/fra/ORA11G/ archivelog/2016_05_04
[oracle@rhel6 2016_05_04]$ ls -lrt
total 1376
-rw-r----- 1 oracle oinstall 1400320 May  4 15:33 o1_mf_1_30_clm9gmg9_.arc
-rw-r----- 1 oracle oinstall    1024 May  4 15:33 o1_mf_1_31_clm9gnh7_.arc
-rw-r----- 1 oracle oinstall    1536 May  4 15:33 o1_mf_1_32_clm9gqsc_.arc
```

注意这几个文件中的序列号 30,31,32，这是我们刚刚完成日志切换之后生成的归档日志。此时，也生成了一些闪回日志：

```
[oracle@rhel6 2016_05_04]$ cd /u01/oracle/fra/ORA11G/flashback/
[oracle@rhel6 flashback]$ ls -lrt
total 1024024
-rw-r----- 1 oracle oinstall 524296192 May  4 15:25 o1_mf_clm8zc1d_.flb
-rw-r----- 1 oracle oinstall 524296192 May  4 15:33 o1_mf_clm8z35k_.flb
```

关于闪回日志的应用，我们在 5.8 一节中探讨。

对于前面初始化参数 db_recovery_file_dest 设置的目录，我们称其为闪回区，或者也称快速恢复区。该区域中，我们可存储控制文件的副本、redo 日志的副本、归档日志以及闪回日志等。当然，如果你对数据库进行备份，那么生成的备份文件也可存放在该目录中。实际上，除了闪回日志必须存放在该目录中之外，其他文件可存放在闪回区中，也可不存放在闪回区中。

Oracle 提供一个 v$recovery_area_usage 视图来监控闪回区的空间利用率：

```
SYS@ora11g> desc v$recovery_area_usage;
 Name                          Null?          Type
 ------------------------- -------------- ---------------------
 FILE_TYPE                                    VARCHAR2(20)
 PERCENT_SPACE_USED                           NUMBER
 PERCENT_SPACE_RECLAIMABLE                    NUMBER
 NUMBER_OF_FILES                              NUMBER
```

```
SYS@ora11g> select sum(PERCENT_SPACE_USED)||'%' from v$recovery_area_usage;
SUM(PERCENT_SPACE_USED)||'%'
-----------------------------------------
24.44%
```

另外，在数据库的 alert 日志中，每次启动时，也会检查闪回区的空间利用情况，并记录下来：

```
SYS@ora11g> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SYS@ora11g> startup;
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size          2257840 bytes
Variable Size       553651280 bytes
Database Buffers    276824064 bytes
Redo Buffers        2371584 bytes
Database mounted.
Database opened.
```

然后查看当前 alert 日志：

```
    [oracle@rhel6 flashback]$ tail -f
/u01/oracle/diag/rdbms/ora11g/ora11g/trace/alert_ora11g.log
Tue May 10 10:32:06 2016
QMNC started with pid=36, OS id=8267
Completed: ALTER DATABASE OPEN
ARC3: Archival started
ARC0: STARTING ARCH PROCESSES COMPLETE
Tue May 10 10:32:06 2016
db_recovery_file_dest_size of 4096 MB is 24.44% used. This is a
user-specified limit on the amount of space that will be used by this
database for recovery-related files, and does not reflect the amount of
space available in the underlying filesystem or ASM diskgroup.
```

注意上述内容的阴影着重显示部分。

5.2 数据库备份实验

为预防数据库出现问题从而导致数据丢失的现象发生，我们需要对数据库进行备份。例如，

某一天数据库用来存储数据的一块磁盘损坏，那么存储在该磁盘上的数据文件都将无法读取。如果想解决这样的问题，我们可以使用恢复，但前提是我们要有备份。本节实验讨论如何对数据库进行备份。

数据库备份可通过多种实现方式，我们可使用 cp 或 dd 这样的操作系统命令来进行数据文件备份，也可以使用 Oracle 自带的 RMAN(Recovery Manager)来进行备份，还可以使用其他第三方的工具(如 NBU)进行备份。甚至还可使用前面 4.6 一节中提到的 exp 或者 expdp 来进行备份。exp 或者 expdp 称为逻辑备份，它并不备份具体数据，而将其转换为一条条 SQL 语句，在恢复数据时重新执行这些 SQL 语句。exp 或者 expdp 这样的命令更多的时候适合用来备份单个表或者多个表，并不适合备份整个数据库，在数据库较大时尤其如此。

使用 cp 或者 dd 这样的操作系统命令来备份，我们通常称为手工备份。在本章的 5.12 一节中，会进行一个对数据库进行完全手工备份的实验。手工备份通常适用于临时对一些数据文件或者表空间进行备份。手工备份较方便，但需要关闭数据库或将需要备份的数据文件置于备份状态。因为在执行备份时，cp 或者 dd 命令并不知道当前是否有用户正在使用这些数据文件。如果要备份的数据文件处于正在被其他用户进行读写的状态，则此时生成的备份可能包含不一致的数据，这样的备份在恢复时是无法使用的。

例如，我们想备份数据库的 tbs_test 表空间中的数据文件：

```
    SYS@ora11g> col file_name for a50
    SYS@ora11g> select file_id,file_name from dba_data_files where
tablespace_name='TBS_TEST';
       FILE_ID FILE_NAME
    ---------- --------------------------------------------------
             5 /u01/oracle/oradata/ora11g/tbs_test_01.dbf
             6 /u01/oracle/oradata/ora11g/tbs_test_02.dbf
```

当数据库处于 open 状态时，可将该表空间置为 backup 状态：

```
    SYS@ora11g> alter tablespace tbs_test begin backup;
    Tablespace altered.
```

然后进行备份：

```
    SYS@ora11g> !cp /u01/oracle/oradata/ora11g/tbs_test_01.dbf
/home/oracle/backup/tbs01.bak;
    SYS@ora11g> !cp /u01/oracle/oradata/ora11g/tbs_test_02.dbf
/home/oracle/backup/tbs02.bak;
```

这里在 sqlplus 中执行操作系统的 cp 命令，需要在 cp 前面加上!。

备份完成后，切记将表空间的状态调整回来：

```
    SYS@ora11g> alter tablespace tbs_test end backup;
```

```
Tablespace altered.
```

当然，也可将数据库关闭，然后使用 cp 或者 dd 命令备份。

在实际环境中，我们更推荐使用 RMAN 对数据库进行备份。该工具为 Oracle 自带，可备份数据文件、表空间、控制文件、参数文件(spfile)、归档日志以及整个数据库。因此 RMAN 是 Oracle 备份恢复中唯一最重要的工具。

接下来使用 RMAN 对数据库进行一些备份实验。

1. RMAN 基本设置

在操作系统中，进入 RMAN 工具非常简单：

```
[oracle@rhel6 ~]$ rman target /
Recovery Manager: Release 11.2.0.4.0 - Production on Wed May 4 16:14:47 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
connected to target database: ORA11G (DBID=11065102)
RMAN>
```

其中，rman target /等价于 rman target sys/oracle@ora11g:

```
RMAN> exit
Recovery Manager complete.
[oracle@rhel6 ~]$ rman target sys/oracle@ora11g
Recovery Manager: Release 11.2.0.4.0 - Production on Wed May 4 16:15:58 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-00554: initialization of internal recovery manager package failed
RMAN-04005: error from target database:
ORA-12154: TNS:could not resolve the connect identifier specified
```

看一下最后的报错信息，Oracle 指出无法解析指定的连接标识符，也就是说前面 rman 命令中写的 ora11g 数据库无法解析。

因为这里的数据库是在第 2 章中采用手工方式创建的，因此很多设置都需要我们手工处理。我们先解决这里的 ORA-12154 错误。

首先编辑 tnsnames.ora 文件：

```
[oracle@rhel6 ~]$ vi $ORACLE_HOME/network/admin/tnsnames.ora
```

在该文件的末尾处添加如下内容：

```
ora11g =
```

```
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = rhel6 )(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = dedicated )
      (SERVICE_NAME = ora11g )
    )
  )
```

这里指定的连接端口为 1521，需要使用默认监听，因此我们启动默认监听：

```
[oracle@rhel6 ~]$ lsnrctl start
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 04-MAY-2016 16:17:27
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Starting /u01/oracle/product/11.2.0/bin/tnslsnr: please wait...
TNSLSNR for Linux: Version 11.2.0.4.0 - Production
System parameter file is
/u01/oracle/product/11.2.0/network/admin/listener.ora
Log messages written to /u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date                04-MAY-2016 16:17:29
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/oracle/product/11.2.0/network/admin/listener.ora
Listener Log File
/u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
The listener supports no services
The command completed successfully
```

此时默认监听上没有注册任何服务，也就是说，如果使用 1521 端口，我们是连接不到数

据库上的。因为前面在进行 4.5 的相关实验时，我们使用了端口为 1526 的 LSNR_2 监听，数据库的服务是注册在该监听上的，我们需要将其改回：

```
SYS@ora11g> show parameter local_listener;
NAME                 TYPE         VALUE
------------------ ----------- ------------------------------
local_listener       string       LSNR_2
SYS@ora11g> alter system set local_listener = '';
System altered.
```

前面说过，等候一分钟，数据库的服务就会自动注册到默认监听上。这里不打算等待，于是手工注册：

```
SYS@ora11g> alter system register;
System altered.
```

然后查看默认监听的状态：

```
[oracle@rhel6 ~]$ lsnrctl status
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 04-MAY-2016 16:17:40
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date                04-MAY-2016 16:17:29
Uptime                    0 days 0 hr. 0 min. 10 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/oracle/product/11.2.0/network/admin/listener.ora
Listener Log File
/u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
Services Summary...
Service "ora11g" has 1 instance(s).
  Instance "ora11g", status READY, has 4 handler(s) for this service...
The command completed successfully
```

此时，数据库的服务已注册到默认监听上，再使用 rman 命令：

```
[oracle@rhel6 ~]$ rman target sys/oracle@ora11g
Recovery Manager: Release 11.2.0.4.0 - Production on Wed May 4 16:17:44 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
connected to target database: ORA11G (DBID=11065102)
RMAN>
```

上述命令中，target 表示我们当前连接的数据库为目标数据库，也就是说我们要执行备份或者恢复操作的数据库。

接下来分析 RMAN 的基本设置：

```
RMAN> show all;
using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name ORA11G are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP OFF; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE FOR
LOAD TRUE ; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/u01/oracle/product/11.2.0/dbs/snapcf_ora11g.f'; # default
```

使用 show all 命令列出当前 RMAN 的所有基本设置。下面将逐一说明。

输出结果中的第一行说明我们当前使用的是目标数据库的控制文件，而非recovery catalog。这句话的意思是，我们当前对 RMAN 进行的设置或者备份操作、这些设置的内容以及生成的备份文件的元数据信息都将存储在目标数据库的控制文件中，而不是 recovery catalog 中。recovery catalog 称为恢复目录，我们将在下一节进行讲解和配置。

控制文件中会存储RMAN的设置以及备份文件的相关信息，前面的4.1一节中已经提到了，读者可以自行翻阅。

上述命令输出中的以 configure 开头的内容显示了当前 RMAN 的默认设置。

RETENTION POLICY 指备份文件的保留策略。Oracle 提供了两种保留策略：基于备份冗

余的保留策略和基于恢复窗口(recovery window)的保留策略。默认为基于冗余的保留策略，也就是说，每次生成的备份文件只需要保留一份副本即可。如果我们想修改该设置，例如想将其改为保留两份副本：

```
RMAN> CONFIGURE RETENTION POLICY TO REDUNDANCY 2;
new RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO REDUNDANCY 2;
new RMAN configuration parameters are successfully stored
```

这样，当以后再进行备份时，Oracle 就会检查生成的备份是否为两个副本，如果不符合这样的保留策略，Oracle 会提示你进行备份。关于这一点，我们稍后进行讲解。

基于恢复窗口(recovery window)的保留策略指的是，如果想将数据库恢复到过去某个时间，那么会需要哪些备份。恢复窗口实际上指数据库可以恢复的时间长度。例如，我们期望所生成的备份可将数据库恢复到过去 30 天中的任意一个时间点，就可以进行如下设置：

```
RMAN> CONFIGURE RETENTION POLICY TO recovery window of 30 days;
old RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO REDUNDANCY 2;
new RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 30 DAYS;
new RMAN configuration parameters are successfully stored
```

这样，当备份之后，我们就可以基于这样的保留策略来检查已经生成的备份，看其能否满足这样的恢复需求。

BACKUP OPTIMIZATION 指 RMAN 备份的自动优化。实际上这个优化非常简单，也就是说，当我们进行备份时，如果有的数据文件已经备份过，并且到这次备份的时候，其内容未曾发生变化，Oracle 就会自动跳过对这些文件的备份。要开启该优化，可执行如下命令：

```
RMAN> CONFIGURE BACKUP OPTIMIZATION on;
new RMAN configuration parameters:
CONFIGURE BACKUP OPTIMIZATION ON;
new RMAN configuration parameters are successfully stored
```

DEFAULT DEVICE TYPE 指默认的备份文件存放设备类型，默认为磁盘。当然，如果使用的是磁带，你也可以修改此设置。这里保持默认。

CONTROLFILE AUTOBACKUP 指控制文件的自动备份。也就是说，如果将该设置调整为 on，则在每次进行 RMAN 备份的时候，Oracle 都会自动备份当前的控制文件和 spfie。鉴于控制文件和参数文件的重要性，我们建议开启此设置：

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP on;
new RMAN configuration parameters:
```

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;
new RMAN configuration parameters are successfully stored
```

CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK，指的是当我们开启控制文件自动备份之后，如果使用的备份设备是磁盘，则其生成的自动备份文件的命名格式应该是怎样的。这里保留默认设置。'%F'的含义可参考官方文档 Database Backup and Recovery Reference 中的 4 RMAN Subclauses 的 formatSpec 部分。

DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET，指的是当我们使用的备份设备类型为磁盘，并且生成的备份为 backupset(备份集)时，备份操作的并行度设置。所谓并行，则指如果当前数据库空闲资源较多，我们在执行一些较大的操作时，可使用多个进程来同时完成。更详细的内容在后续书籍中再做讨论。

关于 backupset，RMAN 备份的时候可生成两种类型的备份：copy(影像副本)和 backupset(备份集)。所谓 copy，就是将要备份的文件直接进行物理复制来生成备份，无论该文件中是否有空的 block，都直接复制。backupset 方式则跳过空的 block 来生成备份文件。因此对于备份集方式生成的备份，备份文件更小。默认情况下，RMAN 进行备份操作时都生成备份集类型的备份。

该设置我们保留默认。

DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK，指的是如果使用的备份设备为磁盘，则生成数据文件的备份时，默认的副本数量为 1。我们保留默认设置。

ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK，指的是如果使用的备份设备为磁盘，则生成归档日志的备份时，默认的副本数量为 1。我们保留默认设置。

MAXSETSIZE，指的是生成的备份集的大小限制。在 RMAN 中，我们每进行一次备份，则生成的所有备份文件的集合称为一个备份集。该设置项用来控制生成的备份集的大小，默认为无限制。我们也保留默认设置。

ENCRYPTION FOR DATABASE，指的是我们是否启用加密备份，对生成的备份文件进行加密；默认不启用加密。我们也保留默认设置。

ENCRYPTION ALGORITHM，指的是如果启用加密，则采用哪种加密算法。Oracle 提供了三种可用的加密算法。

COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE FOR LOAD TRUE，指的是是否启用备份压缩。我们保留默认设置。

ARCHIVELOG DELETION POLICY，指的是归档日志的删除策略。例如，我们可以设置为：完成归档日志到磁盘的一次备份，就可以删除原有的归档日志：

```
RMAN> CONFIGURE ARCHIVELOG DELETION POLICY TO backed up 1 times to disk;
new RMAN configuration parameters:
CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP 1 TIMES TO DISK;
new RMAN configuration parameters are successfully stored
```

实际上，在 DataGuard 环境下讨论归档日志的删除更有意义。

SNAPSHOT CONTROLFILE NAME，最后一个设置项指的是生成控制文件快照时，该快照存放的位置及其名称。因为数据库一旦启动，控制文件便会由 Oracle 数据库一直进行读写操作。如果想备份控制文件，我们需要先对其生成快照，然后备份此快照。该选项我们保留默认设置。

当然，对于 RMAN 的基本设置，如果你在执行了修改之后还想恢复到默认设置，使用 clear 即可。例如，可清除 retention policy 的设置：

```
RMAN> CONFIGURE RETENTION POLICY clear;
old RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 30 DAYS;
RMAN configuration parameters are successfully reset to default value
```

2. 使用 RMAN 进行备份操作

接下来，我们利用 RMAN 对数据库进行备份。

数据文件备份

```
RMAN> backup datafile 1,2,3,5;
Starting backup at 05-MAY-16
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=34 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001 name=/u01/oracle/oradata/ora11g/system01.dbf
input datafile file number=00002 name=/u01/oracle/oradata/ora11g/sysaux01.dbf
input datafile file number=00003
name=/u01/oracle/oradata/ora11g/undotbs01.dbf
input datafile file number=00005
name=/u01/oracle/oradata/ora11g/tbs_test_01.dbf
channel ORA_DISK_1: starting piece 1 at 05-MAY-16
channel ORA_DISK_1: finished piece 1 at 05-MAY-16
piece
handle=/u01/oracle/fra/ORA11G/backupset/2016_05_05/o1_mf_nnndf_TAG20160505T0915
22_clo7obqz_.bkp tag=TAG20160505T091522 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
Finished backup at 05-MAY-16

Starting Control File and SPFILE Autobackup at 05-MAY-16
```

```
piece handle=/u01/oracle/fra/ORA11G/autobackup/2016_05_05/o1_mf_s_
911034937_clo7oswd_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 05-MAY-16
```

如上例所示，我们可以一次性备份一个数据文件，也可以备份多个文件。上面输出中的阴影着重显示部分的 input datafile 表示要备份的文件，handle 表示生成的备份文件存放的位置及其名称。

也需要注意最后一段内容。因为我们前面设置了开启控制文件自动备份，因此这里你可以看到 Oracle 自动对控制文件和 spfile 进行了备份。

默认情况下，这些备份文件都存放在闪回区中，如果你配置了闪回区，另外需要注意的一点是，Oracle 会在闪回区中按照日期来生成不同目录，从而存储文件。因此，读者按这里的例子进行操作时，备份文件的存放位置和书中的位置可能有所不同，需要注意一下日期命名的目录。

要是不想把备份文件存放在闪回区中，而是放在自己指定的位置，可使用 format 命令：

```
RMAN> backup datafile 1,2,3,5 format '/home/oracle/backup/dbf_bak_%U';
```

该命令中的'%U'表示 Oracle 会自动为生成的备份文件指定一个唯一的文件名。和'%F'一样，你也可以参考官方文档 Database Backup and Recovery Reference 中的 4 RMAN Subclauses 的 formatSpec 部分。

备份表空间

```
RMAN> backup tablespace system,sysaux,users;
```

备份控制文件

```
RMAN> backup current controlfile;
```

备份 spfile

```
RMAN> backup spfile;
```

备份归档日志

```
RMAN> backup archivelog all;
```

备份数据库中的所有数据文件

```
RMAN> backup database;
```

一条命令备份数据库所有的数据文件、控制文件、参数文件和日志文件。

```
RMAN> backup database plus archivelog;
```

需要注意，上面这条命令没有写明要备份控制文件和 spfile，但该命令依然会备份这两个文件。原因是，当我们使用 RMAN 进行备份时，只要你备份 1 号数据文件，或者你的备份文件中包含对 1 号文件的备份，Oracle 都会自动触发对控制文件和 spfile 的备份。无论你是否开启控制文件自动备份：

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP clear;
old RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP ON;
RMAN configuration parameters are successfully reset to default value
RMAN> backup datafile 1;
Starting backup at 05-MAY-16
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001 name=/u01/oracle/oradata/ora11g/system01.dbf
channel ORA_DISK_1: starting piece 1 at 05-MAY-16
channel ORA_DISK_1: finished piece 1 at 05-MAY-16
piece
handle=/u01/oracle/fra/ORA11G/backupset/2016_05_05/o1_mf_nnndf_TAG20160505T0930
53_clo8lfcq_.bkp tag=TAG20160505T093053 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 05-MAY-16
channel ORA_DISK_1: finished piece 1 at 05-MAY-16
piece
handle=/u01/oracle/fra/ORA11G/backupset/2016_05_05/o1_mf_ncsnf_TAG20160505T0930
53_clo8lok7_.bkp tag=TAG20160505T093053 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 05-MAY-16
```

注意上述输出内容中的阴影着重显示部分。

另外，上面这些备份都使用 backupset 备份格式，如果想使用 copy 方式的备份，命令如下：

```
RMAN> backup as copy database;
```

3. 备份维护及检查

执行一些 RMAN 的备份操作后，可对这些备份进行检查。

检查我们已经生成了哪些备份文件：

```
RMAN> list backup summary;
List of Backups
===============
Key  TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
---- -- -- - ----------- --------------- ------- ------- ---------- ---
1    B  A  A DISK        04-MAY-16       1       1       NO TAG20160504T171941
2    B  F  A DISK        04-MAY-16       1       1       NO TAG20160504T171943
3    B  F  A DISK        05-MAY-16       1       1       NO TAG20160505T091522
4    B  F  A DISK        05-MAY-16       1       1       NO TAG20160505T091537
5    B  F  A DISK        05-MAY-16       1       1       NO TAG20160505T092404
6    B  F  A DISK        05-MAY-16       1       1       NO TAG20160505T092428
7    B  F  A DISK        05-MAY-16       1       1       NO TAG20160505T092435
8    B  A  A DISK        05-MAY-16       1       1       NO TAG20160505T092526
9    B  F  A DISK        05-MAY-16       1       1       NO TAG20160505T092527
10   B  A  A DISK        05-MAY-16       1       1       NO TAG20160505T092702
11   B  F  A DISK        05-MAY-16       1       1       NO TAG20160505T093053
12   B  F  A DISK        05-MAY-16       1       1       NO TAG20160505T093053
```

查看生成了哪些 copy 方式的备份文件：

```
RMAN> list copy;
```

查看归档日志的备份文件：

```
RMAN> list archivelog all;
```

因为相当一部分备份文件都存放在闪回区中，我们来看一下当前闪回区的空间利用情况：

```
SYS@ora11g> select sum(PERCENT_SPACE_USED),
sum (PERCENT_SPACE_RECLAIMABLE) from v$recovery_area_usage;
SUM(PERCENT_SPACE_USED) SUM(PERCENT_SPACE_RECLAIMABLE)
----------------------- ------------------------------
                  42.69                           1.08
```

percent_space_used表示闪回区的空间使用率，percent_space_reclaimable表示闪回区中已经占用但可以释放的空间比率。我们来删除一些备份：

```
RMAN> delete backup;
using channel ORA_DISK_1
List of Backups Pieces
```

```
    BP Key  BS Key  Pc# Cp# Status      Device Type Piece Name
    ------- ------ --- --- ---------- ----------- ----------
    1       1       1   1   AVAILABLE   DISK   /u01/oracle/fra/ORA11G/backupset/
2016_05_04/o1_mf_annnn_TAG20160504T171941_clmhofw7_.bkp
    2       2       1   1   AVAILABLE   DISK   /u01/oracle/fra/ORA11G/autobackup/
2016_05_04/o1_mf_s_910977583_clmhoh39_.bkp
    3       3       1   1   AVAILABLE   DISK   /u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_nnndf_TAG20160505T091522_clo7obqz_.bkp
    4       4       1   1   AVAILABLE   DISK   /u01/oracle/fra/ORA11G/autobackup/
2016_05_05/o1_mf_s_911034937_clo7oswd_.bkp
    5       5       1   1   AVAILABLE   DISK   /u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_nnndf_TAG20160505T092404_clo85o67_.bkp
    6       6       1   1   AVAILABLE   DISK   /u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_ncnnf_TAG20160505T092428_clo86ftf_.bkp
    7       7       1   1   AVAILABLE   DISK   /u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_nnsnf_TAG20160505T092435_clo86mz3_.bkp
    8       8       1   1   AVAILABLE   DISK   /u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_annnn_TAG20160505T092526_clo8864b_.bkp
    9       9       1   1   AVAILABLE   DISK   /u01/oracle/fra/ORA11G/autobackup/
2016_05_05/o1_mf_s_911035527_clo8878t_.bkp
    10      10      1   1   AVAILABLE   DISK   /u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_annnn_TAG20160505T092702_clo8c6rr_.bkp
    11      11      1   1   AVAILABLE   DISK   /u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_nnndf_TAG20160505T093053_clo8lfcq_.bkp
    12      12      1   1   AVAILABLE   DISK   /u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_ncsnf_TAG20160505T093053_clo8lok7_.bkp

    Do you really want to delete the above objects (enter YES or NO)? yes
```

系统会提示你是否确认要删除这些备份。因为这里生成的备份文件都是进行测试的，所以我们输入 yes 并回车：

```
    deleted backup piece
    backup piece handle=/u01/oracle/fra/ORA11G/backupset/
2016_05_04/o1_mf_annnn_TAG20160504T171941_clmhofw7_.bkp RECID=1 STAMP=910977581
    deleted backup piece
    backup piece handle=/u01/oracle/fra/ORA11G/autobackup/
2016_05_04/o1_mf_s_910977583_clmhoh39_.bkp RECID=2 STAMP=910977583
    deleted backup piece
    backup piece handle=/u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_nnndf_TAG20160505T091522_clo7obqz_.bkp RECID=3 STAMP=911034922
```

```
    deleted backup piece
    backup piece handle=/u01/oracle/fra/ORA11G/autobackup/
2016_05_05/o1_mf_s_911034937_clo7oswd_.bkp RECID=4 STAMP=911034937
    deleted backup piece
    backup piece handle=/u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_nnndf_TAG20160505T092404_clo85o67_.bkp RECID=5 STAMP=911035445
    deleted backup piece
    backup piece handle=/u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_ncnnf_TAG20160505T092428_clo86ftf_.bkp RECID=6 STAMP=911035469
    deleted backup piece
    backup piece handle=/u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_nnsnf_TAG20160505T092435_clo86mz3_.bkp RECID=7 STAMP=911035475
    deleted backup piece
    backup piece
handle=/u01/oracle/fra/ORA11G/backupset/2016_05_05/o1_mf_annnn_TAG20160505T0925
26_clo8864b_.bkp RECID=8 STAMP=911035526
    deleted backup piece
    backup piece handle=/u01/oracle/fra/ORA11G/autobackup/
2016_05_05/o1_mf_s_911035527_clo8878t_.bkp RECID=9 STAMP=911035527
    deleted backup piece
    backup piece handle=/u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_annnn_TAG20160505T092702_clo8c6rr_.bkp RECID=10 STAMP=911035622
    deleted backup piece
    backup piece handle=/u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_nnndf_TAG20160505T093053_clo8lfcq_.bkp RECID=11 STAMP=911035853
    deleted backup piece
    backup piece handle=/u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_ncsnf_TAG20160505T093053_clo8lok7_.bkp RECID=12 STAMP=911035861
    Deleted 12 objects
```

此时再查看闪回区的使用情况：

```
SYS@ora11g> select sum(PERCENT_SPACE_USED),
sum (PERCENT_SPACE_RECLAIMABLE) from v$recovery_area_usage;
SUM(PERCENT_SPACE_USED) SUM(PERCENT_SPACE_RECLAIMABLE)
----------------------- ------------------------------
                  24.55                              0
```

这里在 RMAN 中删除备份，实际上也可删除指定的备份文件。另外，如果你在删除时，不想让 Oracle 提示你进行确认，则可以使用 noprompt：

```
RMAN> delete noprompt backup;
```

当然，如果是在生产系统中，使用这样的不提示方式就要谨慎一些，免得错删文件。如果是使用脚本或者是自己编写的备份维护程序，则可以使用 noprompt，从而实现备份文件的自动化维护。

另外，有时我们会直接在操作系统层面删除备份文件，但此时 Oracle 是不知道的。它只有在用到这些备份文件时，才去检查备份文件是否存在以及是否受损。我们来看下面的示例。

先进行一次备份：

```
    RMAN> backup database format '/home/oracle/backup/dbf_%U';
    Starting backup at 05-MAY-16
    using channel ORA_DISK_1
    channel ORA_DISK_1: starting full datafile backup set
    channel ORA_DISK_1: specifying datafile(s) in backup set
    input datafile file number=00004 name=/u01/oracle/oradata/ora11g/users01.dbf
    input datafile file number=00001 name=/u01/oracle/oradata/ora11g/system01.dbf
    input datafile file number=00002 name=/u01/oracle/oradata/ora11g/sysaux01.dbf
    input datafile file number=00003
name=/u01/oracle/oradata/ora11g/undotbs01.dbf
    input datafile file number=00005
name=/u01/oracle/oradata/ora11g/tbs_test_01.dbf
    input datafile file number=00006
name=/u01/oracle/oradata/ora11g/tbs_test_02.dbf
    channel ORA_DISK_1: starting piece 1 at 05-MAY-16
    channel ORA_DISK_1: finished piece 1 at 05-MAY-16
    piece handle=/home/oracle/backup/dbf_0hr4qkp0_1_1 tag=TAG20160505T095336
comment=NONE
    channel ORA_DISK_1: backup set complete, elapsed time: 00:00:08
    channel ORA_DISK_1: starting full datafile backup set
    channel ORA_DISK_1: specifying datafile(s) in backup set
    including current control file in backup set
    including current SPFILE in backup set
    channel ORA_DISK_1: starting piece 1 at 05-MAY-16
    channel ORA_DISK_1: finished piece 1 at 05-MAY-16
    piece handle=/home/oracle/backup/dbf_0ir4qkp8_1_1 tag=TAG20160505T095336
comment=NONE
    channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
    Finished backup at 05-MAY-16
```

我们去查看一下生成的备份文件：

```
[oracle@rhel6 ~]$ cd backup
[oracle@rhel6 backup]$ ls -lrt
```

```
total 580572
drwxr-xr-x 2 oracle oinstall      4096 May  3 09:36 control
-rw-r----- 1 oracle oinstall 209723392 May  4 16:08 tbs01.bak
-rw-r----- 1 oracle oinstall 104865792 May  4 16:08 tbs02.bak
-rw-r----- 1 oracle oinstall 269852672 May  5 09:53 dbf_0hr4qkp0_1_1
-rw-r----- 1 oracle oinstall  10059776 May  5 09:53 dbf_0ir4qkp8_1_1
```

上述两个标为阴影着重显示部分的文件就是刚生成的备份文件。

第一个文件为当前数据库所有数据文件的备份，因此比较大，第二个备份文件包含控制文件和 spfile。

然后，我们直接删除这两个文件：

```
[oracle@rhel6 backup]$ rm dbf*
```

我们在 RMAN 中查看一下：

```
RMAN> list backup;
List of Backup Sets
===================
BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
13      Full    257.34M    DISK        00:00:04     05-MAY-16
        BP Key: 13   Status: AVAILABLE  Compressed: NO  Tag: TAG20160505T095336
        Piece Name: /home/oracle/backup/dbf_0hr4qkp0_1_1
  List of Datafiles in backup set 13
  File LV   Type    Ckp SCN    Ckp Time   Name
  ---- ---- ------  ---------- ----
  1     Full 465422  05-MAY-16 /u01/oracle/oradata/ora11g/system01.dbf
  2     Full 465422  05-MAY-16 /u01/oracle/oradata/ora11g/sysaux01.dbf
  3     Full 465422  05-MAY-16 /u01/oracle/oradata/ora11g/undotbs01.dbf
  4     Full 465422  05-MAY-16 /u01/oracle/oradata/ora11g/users01.dbf
  5     Full 465422  05-MAY-16 /u01/oracle/oradata/ora11g/tbs_test_01.dbf
  6     Full 465422  05-MAY-16 /u01/oracle/oradata/ora11g/tbs_test_02.dbf

BS Key  Type  LV     Size       Device Type Elapsed Time Completion Time
------- ----  ------ --------   ---------------------------------------
14       Full   9.58M  DISK      00:00:01                 05-MAY-16
        BP Key: 14   Status: AVAILABLE  Compressed: NO  Tag: TAG20160505T095336
        Piece Name: /home/oracle/backup/dbf_0ir4qkp8_1_1
  SPFILE Included: Modification time: 05-MAY-16
```

```
  SPFILE db_unique_name: ORA11G
  Control File Included: Ckp SCN: 465425      Ckp time: 05-MAY-16
```

读者可以认真看一下上述命令的输出内容。可看到这些备份文件中分别包含了对哪些文件的备份。但是我们这里的问题是，这两个备份文件已经被删除了，但是 Oracle 显然还不知道。因为上述输出内容中显示这两个文件的状态是 AVAILABLE！

显然，我们需要让 Oracle 知道这件事情：

```
RMAN> crosscheck backup;
using channel ORA_DISK_1
crosschecked backup piece: found to be 'EXPIRED'
backup piece handle=/home/oracle/backup/dbf_0hr4qkp0_1_1 RECID=13
STAMP=911037216
crosschecked backup piece: found to be 'EXPIRED'
backup piece handle=/home/oracle/backup/dbf_0ir4qkp8_1_1 RECID=14
STAMP=911037225
Crosschecked 2 objects
```

这里用到的命令为 crosscheck，该命令会对 RMAN 已经记录的备份文件的信息和实际的备份文件进行交叉检验，如果实际上备份文件已经不存在，则这些文件在 RMAN 中会被标记为 expired：

```
RMAN> list expired backup;
List of Backup Sets
===================
BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
13      Full    257.34M    DISK        00:00:04     05-MAY-16
        BP Key: 13   Status: EXPIRED  Compressed: NO  Tag: TAG20160505T095336
        Piece Name: /home/oracle/backup/dbf_0hr4qkp0_1_1
  List of Datafiles in backup set 13
  File LV    Type      Ckp SCN    Ckp Time  Name
  ---- ---- -------- ---------- ----
  1     Full 465422    05-MAY-16 /u01/oracle/oradata/ora11g/system01.dbf
  2     Full 465422    05-MAY-16 /u01/oracle/oradata/ora11g/sysaux01.dbf
  3     Full 465422    05-MAY-16 /u01/oracle/oradata/ora11g/undotbs01.dbf
  4     Full 465422    05-MAY-16 /u01/oracle/oradata/ora11g/users01.dbf
  5     Full 465422    05-MAY-16 /u01/oracle/oradata/ora11g/tbs_test_01.dbf
  6     Full 465422    05-MAY-16 /u01/oracle/oradata/ora11g/tbs_test_02.dbf
```

```
BS Key Type  LV      Size      Device Type Elapsed Time  Completion Time
------- ----- -------- -------- ------------------------ ----------------
14      Full  9.58M   DISK      00:00:01                  05-MAY-16
        BP Key: 14   Status: EXPIRED  Compressed: NO  Tag: TAG20160505T095336
        Piece Name: /home/oracle/backup/dbf_0ir4qkp8_1_1
  SPFILE Included: Modification time: 05-MAY-16
  SPFILE db_unique_name: ORA11G
  Control File Included: Ckp SCN: 465425      Ckp time: 05-MAY-16
```

显然，这样的备份文件信息是需要从 RMAN 中删除的：

```
RMAN> delete noprompt expired backup;
using channel ORA_DISK_1
List of Backup Pieces
BP Key BS Key Pc# Cp# Status   Device Type Piece Name
------- ------- --- --- -------- ----------- ------------------------------
13      13       1   1   EXPIRED  DISK    /home/oracle/backup/dbf_0hr4qkp0_1_1
14      14       1   1   EXPIRED  DISK    /home/oracle/backup/dbf_0ir4qkp8_1_1
deleted backup piece
backup piece handle=/home/oracle/backup/dbf_0hr4qkp0_1_1 RECID=13
STAMP=911037216
deleted backup piece
backup piece handle=/home/oracle/backup/dbf_0ir4qkp8_1_1 RECID=14
STAMP=911037225
Deleted 2 EXPIRED objects
```

除了 list 这条查看生成的备份文件相关信息的命令外，还有一个 report 命令：

```
RMAN> report schema;
Report of database schema for database with db_unique_name ORA11G
List of Permanent Datafiles
File Size(MB) Tablespace RB segs Datafile Name
---- -------- ----------- ------- ----------------------------------------
1    325      SYSTEM      ***     /u01/oracle/oradata/ora11g/system01.dbf
2    325      SYSAUX      ***     /u01/oracle/oradata/ora11g/sysaux01.dbf
3    278      UNDOTBS1    ***     /u01/oracle/oradata/ora11g/undotbs01.dbf
4    500      USERS       ***     /u01/oracle/oradata/ora11g/users01.dbf
5    200      TBS_TEST    ***     /u01/oracle/oradata/ora11g/tbs_test_01.dbf
6    100      TBS_TEST    ***     /u01/oracle/oradata/ora11g/tbs_test_02.dbf

List of Temporary Files
=======================
```

```
File Size(MB) Tablespace         Maxsize(MB) Tempfile Name
---  --------- ---------------------------- --------------------
1    72        TEMPTS1     10240   /u01/oracle/oradata/ora11g/temp01.dbf
2    100       TEMP001       100   /u01/oracle/oradata/ora11g/temp001.dbf
3    100       TEMP002       100   /home/oracle/temp002.dbf
```

可使用上述命令来查看目标数据库的数据文件和临时文件信息。当然，report 的另一个重要用途是，我们可以基于 retention policy，查看还有哪些文件需要备份：

```
RMAN> CONFIGURE RETENTION POLICY TO recovery window of 30 days;
new RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 30 DAYS;
new RMAN configuration parameters are successfully stored
RMAN>report need backup;
RMAN retention policy will be applied to the command
RMAN retention policy is set to recovery window of 30 days
Report of files that must be backed up to satisfy 30 days recovery window
File Days  Name
---- ----- -----------------------------------------------------
```

上条命令没有输出文件，显示基于 30 天恢复窗口的备份保留策略，并没有需要备份的文件。在实际系统中，如果有，则需要进行备份。

那么，基于当前的备份保留策略，如果有些备份文件已经不需要了，我们该如何发现这些文件？

```
RMAN> report obsolete;
RMAN retention policy will be applied to the command
RMAN retention policy is set to recovery window of 30 days
no obsolete backups found
```

使用上述命令即可。如果发现有输出内容，则可删除这些备份文件：

```
RMAN> delete obsolete;
```

关于使用 RMAN，还有一些小技巧。这里稍微提及两个。第一，就是在使用 RMAN 执行命令时，RMAN 的输出信息中并不包含该命令，并且执行时间也只显示年月日，而没有更详细的信息。我们可以稍加设置：

```
[oracle@rhel6 ~]$ export NLS_DATE_FORMAT='YYYY-MM-DD HH24:MI:SS'
```

先在操作系统级别设置环境变量，用来控制日期时间类型的数值的显示格式。然后在 RMAN 中执行如下命令：

```
[oracle@rhel6 ~]$ rman target /
Recovery Manager: Release 11.2.0.4.0 - Production on Thu May 5 10:21:34 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
connected to target database: ORA11G (DBID=11065102)
RMAN>set echo on;
echo set on
RMAN> backup database;
backup database;
Starting backup at 2016-05-05 10:21:42
......
```

此处省略后面的输出。这样设置后，我们就可以在执行命令之后的输出信息中，查看到我们执行的具体命令，以及执行该命令的详细时间。

另一个技巧就是 Oracle 提供了一个 v$rman_output 视图，用来存储曾经执行过的 RMAN 命令对应的输出信息：

```
SYS@ora11g> select OUTPUT from v$rman_output;
.......
OUTPUT
--------------------------------------------------------------------------
echo set on
backup database;
Starting backup at 2016-05-05 10:21:42
......
```

我们省略掉其他内容，可以看到刚执行的 RMAN 命令的输出信息。因此，可使用这个视图来查看曾执行过哪些 RMAN 操作。但要注意，该视图只存储 32 768 行记录。

5.3　recovery catalog 配置实验

前一节在对 RMAN 进行基本配置时提到过，默认情况下，我们使用目标数据库的控制文件来存储已经生成的备份文件的元数据信息。那么问题来了，如果目标数据库的控制文件丢了或者损坏了，Oracle 岂不就不知道当前数据库有哪些可用的备份文件了？

因此，如果可以把所生成的备份文件的元数据信息存放到其他地方而非目标数据库中，就可以避免上述问题。Oracle 提供的解决方法是使用 recovery catalog(恢复目录)。我们可在其他数据库中创建这样一个恢复目录，用来存储目标数据库的备份文件的元数据信息。当然，我们要先再创建一个数据库。

在第 2 章中，我们用手工方式创建了 ora11g 数据库，接下来，我们使用 dbca 这个图形化工具来创建新数据库。

首先启动 dbca 工具：

```
[oracle@rhel6 ~]$ exit
logout
You have new mail in /var/spool/mail/root
[root@rhel6 Desktop]# xhost +
access control disabled, clients can connect from any host
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ dbca
```

将弹出图 5-1 所示窗口：



图 5-1　欢迎页面

在图 5-2 中点击 Next：

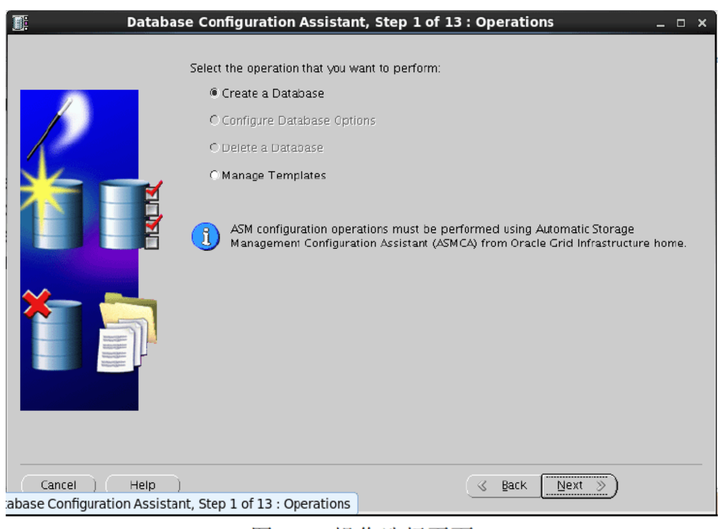


图 5-2　操作选择页面

这里需要创建一个数据库，因此保持默认，在图 5-3 中点击 Next：

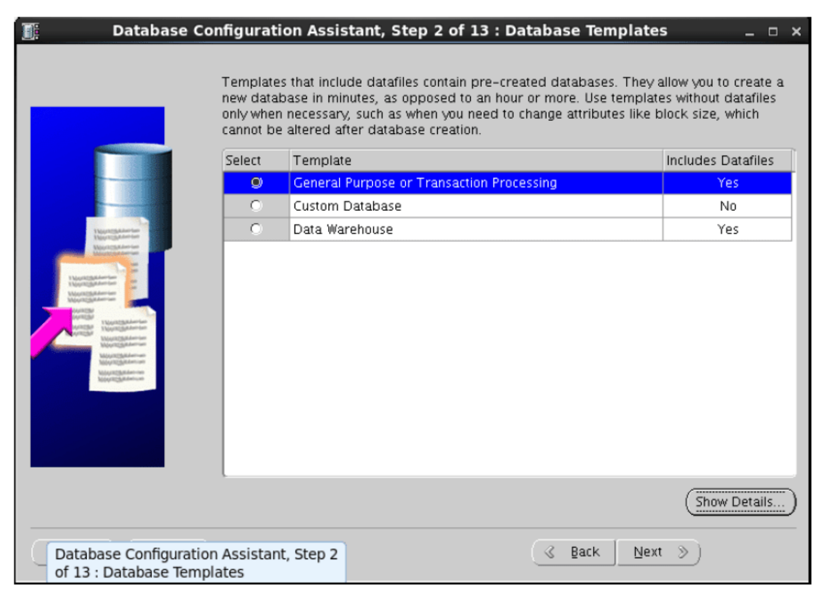


图 5-3　数据库模板选择页面

这里需要选择数据库的用途，保持默认，在图 5-4 中点击 Next：

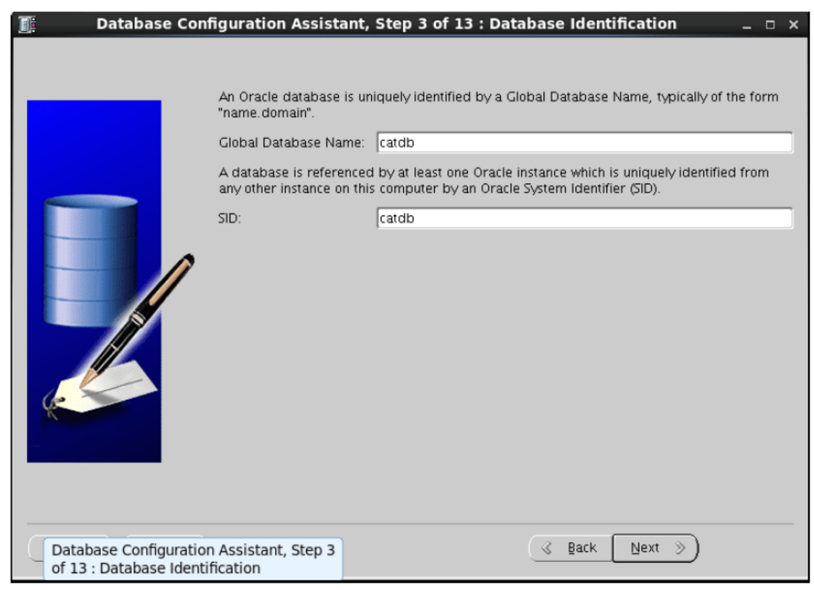


图 5-4　数据库名称设置页面

输入数据库名 catdb，然后点击 Next(图 5-5)：

选择是否配置 EM。EM 是 Oracle 提供的一个对数据库进行监控和管理的图形化工具，稍后会介绍一下这个工具。保持默认，点击 Next(图 5-6)：

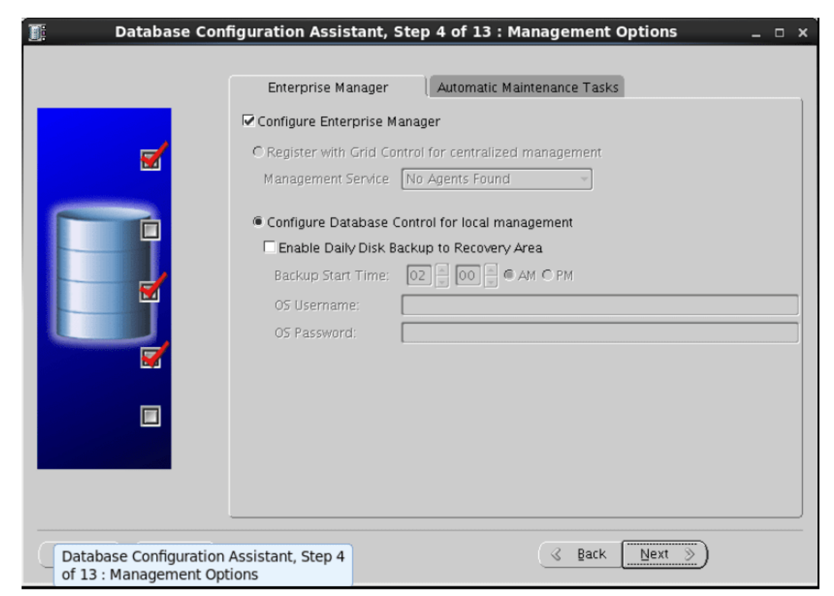


图 5-5　是否配置 EM 页面

弹出如上的对话窗口，指出如要配置 EM，需要启动监听。

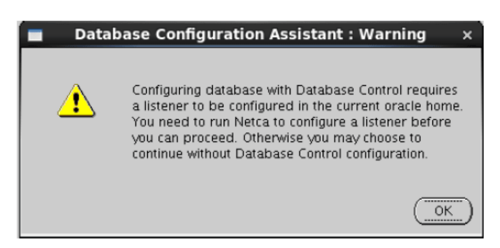


图 5-6　告警页面

我们打开一个新窗口，启动默认监听：

```
[oracle@rhel6 ~]$ lsnrctl start
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 05-MAY-2016 10:46:25
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Starting /u01/oracle/product/11.2.0/bin/tnslsnr: please wait...
TNSLSNR for Linux: Version 11.2.0.4.0 - Production
System parameter file is
/u01/oracle/product/11.2.0/network/admin/listener.ora
Log messages written to /u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
STATUS of the LISTENER
------------------------
```

```
    Alias                     LISTENER
    Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
    Start Date                05-MAY-2016 10:46:26
    Uptime                    0 days 0 hr. 0 min. 0 sec
    Trace Level               off
    Security                  ON: Local OS Authentication
    SNMP                      OFF
    Listener Parameter File
/u01/oracle/product/11.2.0/network/admin/listener.ora
    Listener Log File
/u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
    Listening Endpoints Summary...
      (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
      (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
    The listener supports no services
    The command completed successfully
```

然后点击对话窗口中的 OK，继续点击 Next(图 5-7)：

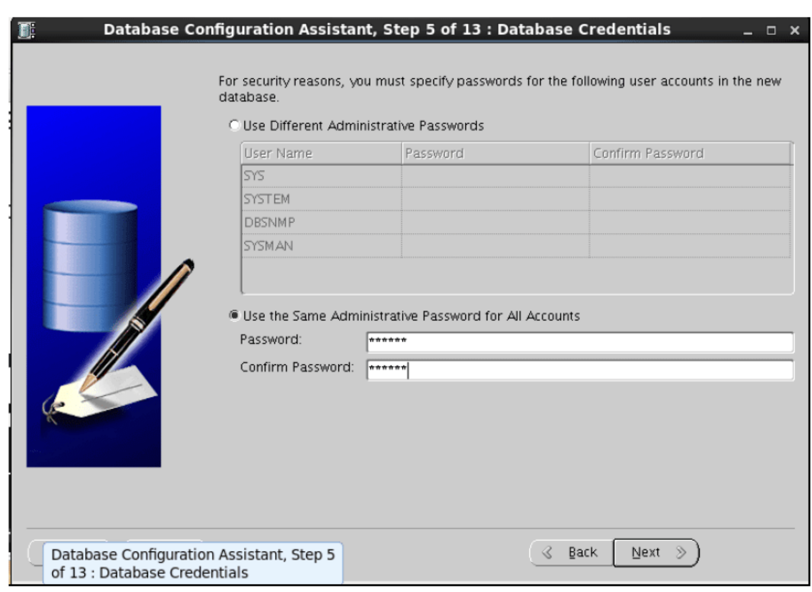


图 5-7　密码设置页面

这里要求设置几个数据库默认用户的密码，我们选中 Use the Same Administrative Password for All Accounts，然后输入 oracle，点击 Next(图 5-8)：

弹出一个对话框，提示所设置的密码不符合 Oracle 推荐的密码复杂度策略，我们点击 Yes，然后继续点击 Next(图 5-9)：

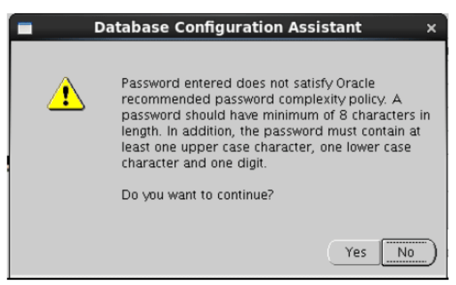


图 5-8 密码复杂度告警页面

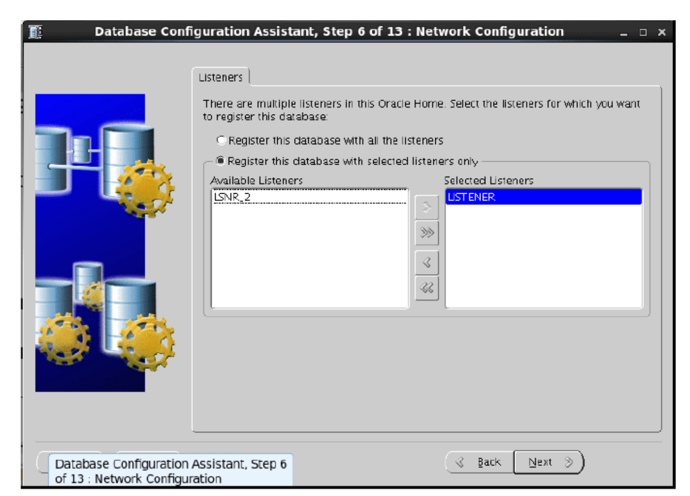


图 5-9 监听选择页面

这里选择要把数据库的服务注册到哪个监听上，我们选择 LISTENER，然后点击 Next(图 5-10)：

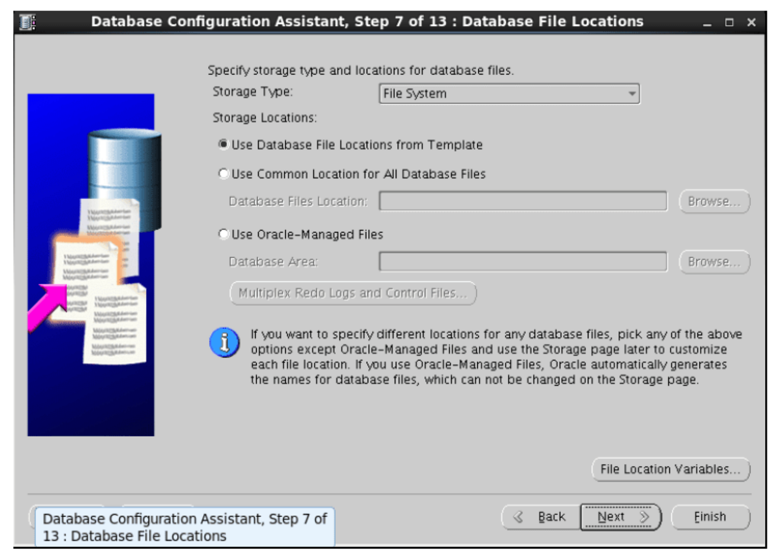


图 5-10 存储格式选择页面

选择文件系统，保持默认，点击 Next(图 5-11)：

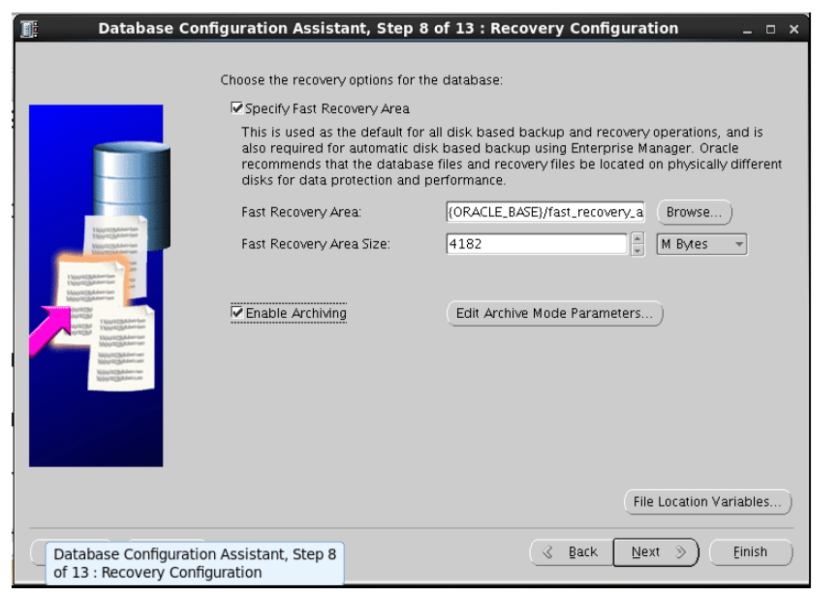


图 5-11　闪回及归档设置页面

对于闪回区设置，我们勾选 Enable Archiving，启用归档，然后点击 Next(图 5-12)。

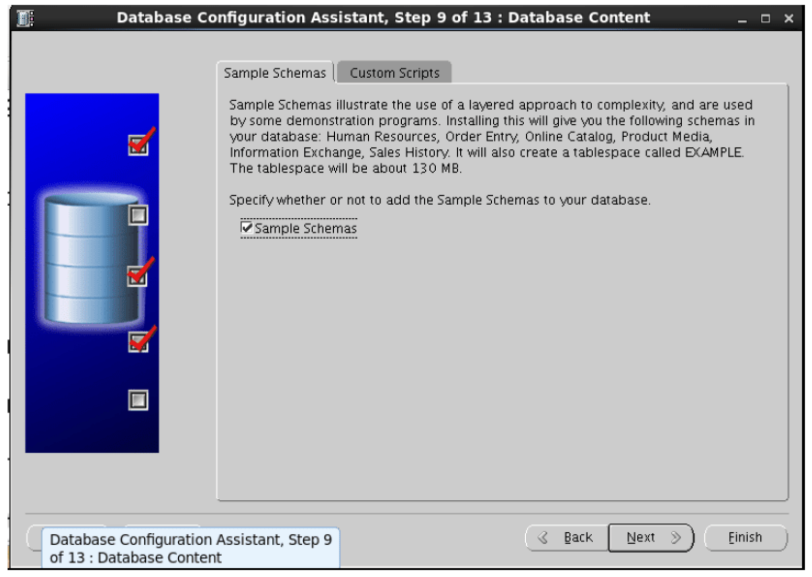


图 5-12　选择创建样例数据页面

对于是否创建样例数据，我们勾选 Sample Schemas，点击 Next(图 5-13)。

该页面共有四个标签，分别用来设置内存、processes、字符集以及数据库连接模式，我们将总内存值设置为 800，其他均保持默认，然后点击 Next(图 5-14)。

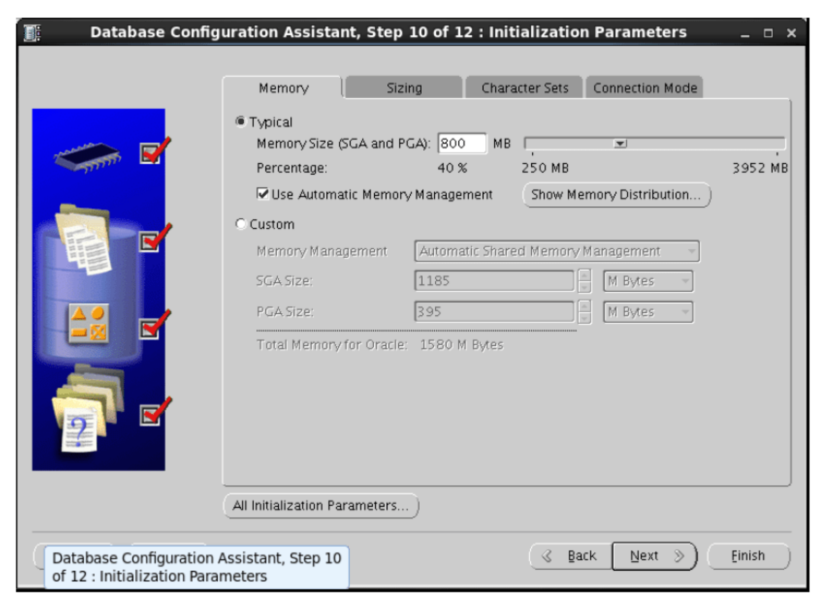


图 5-13　初始化设置页面

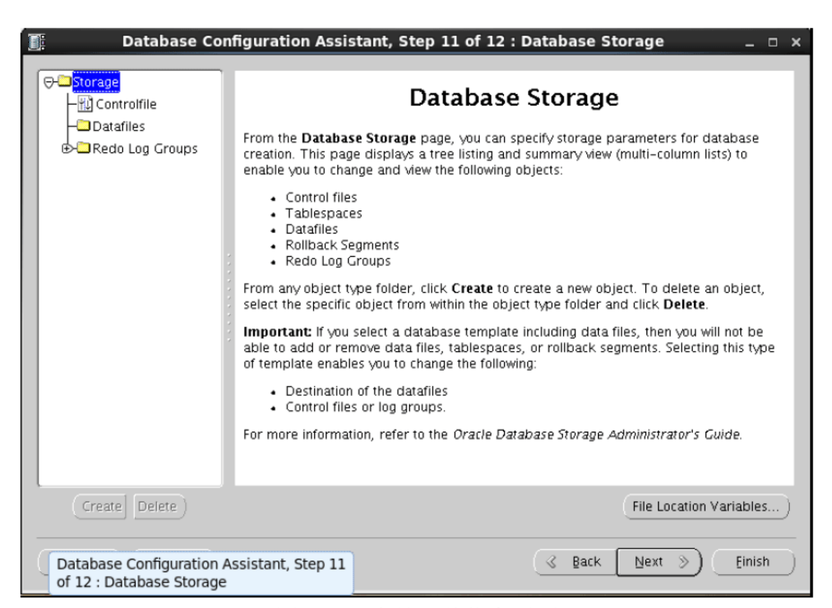


图 5-14　存储汇总信息页面

弹出当前数据库创建存储方面的汇总信息，保持默认，点击 Next(图 5-15)。

继续保持默认，点击 Finish(图 5-16)：

弹出数据库创建的汇总信息，点击 OK，进入数据库创建进度页面(图 5-17)。

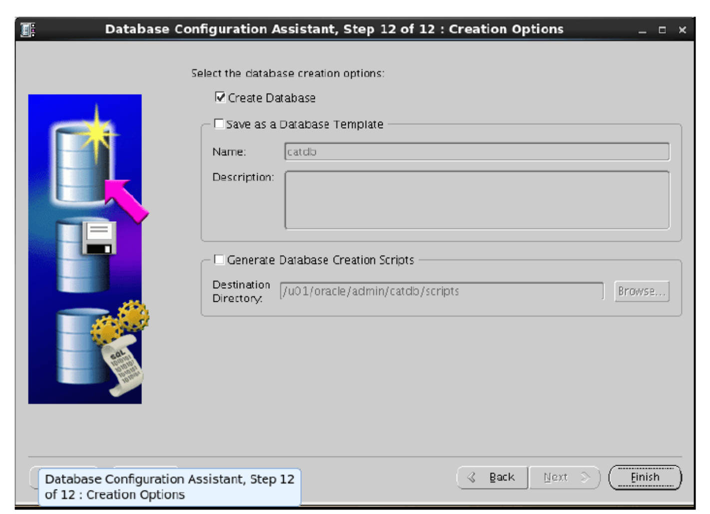


图 5-15　数据库创建页面

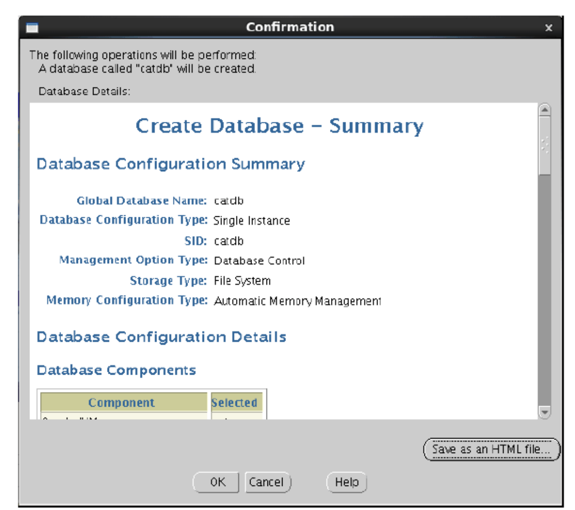


图 5-16　数据库创建信息汇总页面

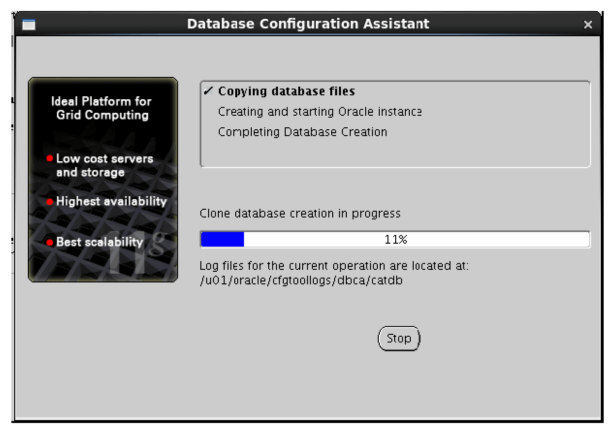


图 5-17　数据库创建进度页面

待数据库创建完成，弹出如图 5-18 所示的对话框。

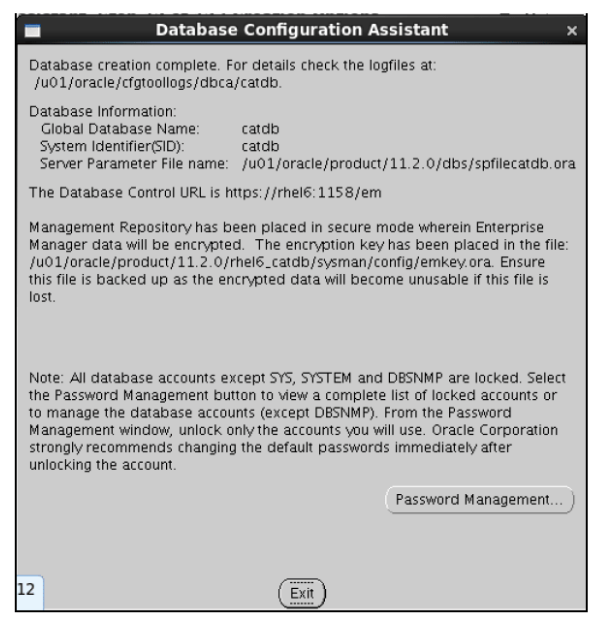


图 5-18 数据库创建完成信息页面

点击 Exit，数据库创建完毕。

然后，我们来登录一下该数据库：

```
[oracle@rhel6 ~]$ export ORACLE_SID=catdb
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Thu May 5 11:04:09 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SYS@catdb>
```

这样，catdb 数据库就准备完毕了。

接下来来配置 recovery catalog。

1) 在 catdb 中创建存储 recovery catalog 的表空间。

```
SYS@catdb> create tablespace tbs_rc datafile
'/u01/oracle/oradata/catdb/tbs_rc_01.dbf' size 100M;
Tablespace created.
```

2) 创建 recovery catalog 管理用户并授权。

```
SYS@catdb> create user u_rcadm identified by admin default tablespace tbs_rc;
```

```
User created.
SYS@catdb> grant connect,resource,recovery_catalog_owner to u_rcadm;
Grant succeeded.
```

前文已经提过 resource 和 connect 两个角色。recovery_catalog_owner 角色包含的权限如下：

```
SYS@catdb> select privilege from role_sys_privs where
role='RECOVERY_CATALOG_OWNER';
PRIVILEGE
----------------------------------------
CREATE SYNONYM
CREATE CLUSTER
ALTER SESSION
CREATE DATABASE LINK
CREATE SESSION
CREATE TABLE
CREATE SEQUENCE
CREATE PROCEDURE
CREATE VIEW
CREATE TYPE
CREATE TRIGGER

11 rows selected.
```

3) 创建 recovery catalog

```
[oracle@rhel6 ~]$ rman catalog u_rcadm/admin@catdb;
Recovery Manager: Release 11.2.0.4.0 - Production on Thu May 5 14:25:59 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-00554: initialization of internal recovery manager package failed
RMAN-04004: error from recovery catalog database: ORA-12541: TNS:no listener
```

错误显示当前的监听没有启动，这里启动默认监听：

```
[oracle@rhel6 ~]$lsnrctl start
```

然后等候一分钟，再查看：

```
 [oracle@rhel6 ~]$ lsnrctl services
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 05-MAY-2016 14:33:29
```

```
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
Services Summary...
Service "catdb" has 1 instance(s).
  Instance "catdb", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "catdbXDB" has 1 instance(s).
  Instance "catdb", status READY, has 1 handler(s) for this service...
    Handler(s):
      "D000" established:0 refused:0 current:0 max:1022 state:ready
         DISPATCHER <machine: rhel6.oracle.com, pid: 3836>
         (ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=11903))
Service "ora11g" has 1 instance(s).
  Instance "ora11g", status READY, has 4 handler(s) for this service...
    Handler(s):
      "D002" established:0 refused:0 current:0 max:1022 state:ready
         DISPATCHER <machine: rhel6.oracle.com, pid: 3703>
         (ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=20590))
      "D001" established:0 refused:0 current:0 max:1022 state:ready
         DISPATCHER <machine: rhel6.oracle.com, pid: 3701>
         (ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=43753))
      "D000" established:0 refused:0 current:0 max:1022 state:ready
         DISPATCHER <machine: rhel6.oracle.com, pid: 3699>
         (ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=49802))
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
The command completed successfully
```

可见，此时 ora11g 和 catdb 两个数据库的服务都注册到默认监听了。

接下来连接到 RMAN：

```
[oracle@rhel6 ~]$ rman catalog u_rcadm/admin@catdb;
Recovery Manager: Release 11.2.0.4.0 - Production on Thu May 5 14:35:52 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
connected to recovery catalog database
RMAN> create catalog;
recovery catalog created
```

这样就创建好了 recovery catalog。此时查看 u_rcadm 用户，可以发现该用户多了很多对象。我们还要把目标数据库 ora11g 注册到该恢复目录中：

```
RMAN> connect target sys/oracle@ora11g
connected to target database: ORA11G (DBID=11065102)
RMAN> register database;
database registered in recovery catalog
starting full resync of recovery catalog
full resync complete
```

这样就完成数据库注册了。查看 u_rcadmin 下的表可以确认：

```
SYS@catdb> conn u_rcadm/admin;
Connected.
U_RCADM@catdb> select * from rc_database;
    DB_KEY  DBINC_KEY  DBID         NAME      RESETLOGS_CHANGE#  RESETLOGS
---------- ---------- ----------  --------- ------------------ -------
         1          2  11065102   ORA11G                     1  22-APR-16
```

以后，我们再对 ora11g 进行备份操作时，就可将备份文件的元数据信息写到 catdb 的恢复目录中。不过连接 rman 时，就需要这样写了：

```
[oracle@rhel6 ~]$ rman target sys/oracle@ora11g catalog u_rcadm/admin@catdb
Recovery Manager: Release 11.2.0.4.0 - Production on Thu May 5 14:41:05 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
connected to target database: ORA11G (DBID=11065102)
connected to recovery catalog database
RMAN>
```

需要注意，我们只是将这些备份文件的元数据信息存放到恢复目录中。实际备份文件依在原位置。

这里还有一个细节问题，如果我们在使用 RMAN 进行备份操作时，只连接了目标数据库，但忘记连接 catalog 数据库，那么这些备份生成的备份文件的元数据信息显然就不在 catalog 中了。此时，可手工将这些备份文件的信息登记到 catalog 中。例如，前面 5.2 一节中对表空间 tbs_test 的两个数据文件用 cp 命令所做的备份可以登记到 catalog 中：

```
RMAN> catalog datafilecopy '/home/oracle/backup/tbs01.bak';
cataloged datafile copy
datafile copy file name=/home/oracle/backup/tbs01.bak RECID=3 STAMP=911473448
RMAN> catalog datafilecopy '/home/oracle/backup/tbs02.bak';
cataloged datafile copy
datafile copy file name=/home/oracle/backup/tbs02.bak RECID=4 STAMP=911473457
```

当然，使用恢复目录还有其他优势，比如可存储 rman 脚本、保存更多备份信息等。

另外需要注意，如果使用恢复目录，则需要同时打开目标数据库和 catalog 数据库，并且监听也需要启动才行，然后使用 rman 来同时连接这两个数据库。

至此，我们已经完成数据库备份的基本配置和相关操作的实验，接下来，可利用数据库备份进行一些恢复实验。这些恢复实验都是实际生产环境中比较有代表性的实验，希望读者能认真掌握。

5.4 参数文件丢失实验

在前面 4.7 一节的数据库启动的三个阶段中，我们提到了数据库启动第一步要使用的文件，就是参数文件。默认先查找 spfile，如果没有，再查找 pfile，如果 pfile 也找不到，则直接报错。所以实际情况是，只要 spfile 和 pfile 二者存在其一，数据库就能正常启动。这里以两个参数文件均丢失为例，来讲解参数文件的恢复。

首先对 ora11g 数据库做一次完全备份：

```
[oracle@rhel6 ~]$  rman target sys/oracle@ora11g catalog u_rcadm/admin@catdb
Recovery Manager: Release 11.2.0.4.0 - Production on Tue May 10 11:06:36 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
connected to target database: ORA11G (DBID=12479381)
connected to recovery catalog database
RMAN> backup database plus archivelog;
```

该命令输出结果太长，我们直接省略。接下来删除 ora11g 的 spfile 和 pfile：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ cd $ORACLE_HOME/dbs
[oracle@rhel6 dbs]$ ls
hc_catdb.dat   initora11g.ora orapwcatdb     spfilecatdb.ora
hc_ora11g.dat  lkCATDB        orapwora11g    spfileora11g.ora
init.ora       lkORA11G       snapcf_ora11g.f
[oracle@rhel6 dbs]$ rm spfileora11g.ora
[oracle@rhel6 dbs]$ rm initora11g.ora
```

然后重启数据库：

```
SYS@ora11g> startup force;
ORA-01078: failure in processing system parameters
LRM-00109: could not open parameter file
'/u01/oracle/product/11.2.0/dbs/initora11g.ora'
```

显示无法找到 pfile 文件，因此无法启动。

我们使用 RMAN 执行恢复，先连接到 RMAN：

```
[oracle@rhel6 ~]$ rman target sys/oracle@ora11g catalog u_rcadm/admin@catdb
Recovery Manager: Release 11.2.0.4.0 - Production on Thu May 5 15:02:47 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-00554: initialization of internal recovery manager package failed
RMAN-04005: error from target database:
ORA-12514: TNS:listener does not currently know of service requested in connect
descriptor
```

这里报错，我们查看当前的监听状态：

```
[oracle@rhel6 dbs]$ lsnrctl status
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 05-MAY-2016 15:04:26
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date                05-MAY-2016 14:32:00
Uptime                    0 days 0 hr. 32 min. 25 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/oracle/product/11.2.0/network/admin/listener.ora
Listener Log File
/u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
Services Summary...
Service "catdb" has 1 instance(s).
  Instance "catdb", status READY, has 1 handler(s) for this service...
Service "catdbXDB" has 1 instance(s).
  Instance "catdb", status READY, has 1 handler(s) for this service...
```

```
The command completed successfully
```

可以看到，因为当前 ora11g 数据库启动的第一步就因为找不到参数文件而报错，数据库无法启动，PMON 进程无法工作，也就无法完成数据库服务的注册。也就是说，我们无法通过监听连接到 ora11g，于是改用 IPC 连接方式：

```
[oracle@rhel6 ~]$ rman target / catalog u_rcadm/admin@catdb
Recovery Manager: Release 11.2.0.4.0 - Production on Thu May 5 15:03:06 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
connected to target database (not started)
connected to recovery catalog database
RMAN>
```

当然，即便数据库无法正常启动，我们也有办法将数据库的服务注册到监听上，因为还有一种注册方法称为“静态注册”。我们将在后续书籍进行讲述，这里暂且略过。

接下来，首先强制启动实例，不然没有进程来执行操作了：

```
RMAN> startup nomount force;
startup failed: ORA-01078: failure in processing system parameters
LRM-00109: could not open parameter file
'/u01/oracle/product/11.2.0/dbs/initora11g.ora'
starting Oracle instance without parameter file for retrieval of spfile
Oracle instance started
Total System Global Area   1068937216 bytes
Fixed Size                 2260088 bytes
Variable Size              281019272 bytes
Database Buffers           780140544 bytes
Redo Buffers               5517312 bytes
```

然后，从自动备份中还原参数文件：

```
RMAN>restore spfile from autobackup;
Starting restore at 05-MAY-16
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=19 device type=DISK
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160505
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160504
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160503
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160502
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160501
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160430
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160429
```

```
channel ORA_DISK_1: no AUTOBACKUP in 7 days found
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of restore command at 05/05/2016 15:09:22
RMAN-06172: no AUTOBACKUP found or specified handle is not a valid copy or piece
```

报错说找不到可用的自动备份，我们来看一下 RMAN 的控制文件自动备份设置：

```
RMAN> show controlfile autobackup;
RMAN configuration parameters for database with db_unique_name DUMMY are:
CONFIGURE CONTROLFILE AUTOBACKUP OFF; # default
```

原来是我们备份之前忘记了开启，那我们换一种方式，找到包含参数文件的备份文件：

```
[oracle@rhel6 ~]$ cd /u01/oracle/fra/ORA11G/backupset/2016_05_05
[oracle@rhel6 2016_05_05]$ ls -lrt
total 549604
-rw-r----- 1 oracle oinstall 270368768 May  5 10:21
o1_mf_nnndf_TAG20160505T102143_clockqf1_.bkp
-rw-r----- 1 oracle oinstall  11646976 May  5 14:49
o1_mf_annnn_TAG20160505T144916_clov7d8s_.bkp
-rw-r----- 1 oracle oinstall 270712832 May  5 14:49
o1_mf_nnndf_TAG20160505T144917_clov7g8v_.bkp
-rw-r----- 1 oracle oinstall  10059776 May  5 14:49
o1_mf_ncsnf_TAG20160505T144917_clov7yf2_.bkp
-rw-r----- 1 oracle oinstall      3072 May  5 14:49
o1_mf_annnn_TAG20160505T144936_clov80cx_.bkp
```

我们在本节实验开始时进行的备份操作生成的文件就是这些。注意，这些文件的名称是 Oracle 自动生成的，因此各位读者生成的备份文件与这里的名称应该是不同的。那么，哪个包含参数文件呢？我们知道参数文件和控制文件与数据文件相比，往往都要小一些，因此我们从下到上，从小到大来尝试一下。我们先尝试最后一个文件：

```
RMAN> restore spfile
from
'/u01/oracle/fra/ORA11G/backupset/2016_05_05/o1_mf_annnn_TAG20160505T144936_clo
v80cx_.bkp';
Starting restore at 05-MAY-16
using channel ORA_DISK_1
channel ORA_DISK_1: restoring spfile from AUTOBACKUP
/u01/oracle/fra/ORA11G/backupset/2016_05_05/o1_mf_annnn_TAG20160505T144936_clov
```

```
80cx_.bkp
    RMAN-00571: ===========================================================
    RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
    RMAN-00571: ===========================================================
    RMAN-03002: failure of restore command at 05/05/2016 15:16:33
    ORA-19870: error while restoring backup piece
/u01/oracle/fra/ORA11G/backupset/2016_05_05/o1_mf_annnn_TAG20160505T144936_clov
80cx_.bkp
    ORA-19626: backup set type is archived log - can not be processed by this
conversation
```

再次报错，显示该备份文件中包含的是归档日志，好吧，我们再尝试其他文件：

```
    RMAN> restore spfile
    from '/u01/oracle/fra/ORA11G/backupset/
2016_05_05/o1_mf_ncsnf_TAG20160505T144917_clov7yf2_.bkp';
    Starting restore at 05-MAY-16
    using channel ORA_DISK_1
    channel ORA_DISK_1: restoring spfile from AUTOBACKUP /u01/oracle/fra/ORA11G/
backupset/2016_05_05/o1_mf_ncsnf_TAG20160505T144917_clov7yf2_.bkp
    channel ORA_DISK_1: SPFILE restore from AUTOBACKUP complete
    Finished restore at 05-MAY-16
```

这一次正确无误。因为我们这个实验只是参数文件丢失，所以只要搞定 spfile，就可以重启数据库了：

```
    RMAN> startup force;
    Oracle instance started
    database mounted
    database opened
    Total System Global Area    835104768 bytes
    Fixed Size                  2257840 bytes
    Variable Size               553651280 bytes
    Database Buffers            276824064 bytes
    Redo Buffers                2371584 bytes
```

然后检查一下数据库的状态：

```
    SYS@ora11g> select status from v$instance;
    select status from v$instance
    *
    ERROR at line 1:
```

```
ORA-01034: ORACLE not available
Process ID: 0
Session ID: 1 Serial number: 5
SYS@ora11g> exit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
[oracle@rhel6 dbs]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Tue May 10 11:18:26 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SYS@ora11g> select status from v$instance;
STATUS
------------
OPEN
```

5.5　控制文件恢复实验

在前面的 4.1 一节，我们已经完成了控制文件的多路复用。将当前数据库的控制文件从两个变成三个。接下来，如果控制文件丢失了，又该如何处理？这里有两种情况：第一种，三个控制文件丢失了一个或者两个，至少还有一个完好；第二种，三个控制文件都丢失，但在丢失之前 RMAN 已经有备份。

先分析第一种情况：

```
SYS@ora11g> show parameter control_files;
NAME           TYPE        VALUE
-------------- ----------- ------------------------------
control_files  string      /u01/oracle/oradata/ora11g/con
               trol01.ctl, /u01/oracle/fra/co
               ntrol2.ctl, /home/oracle/backu
                           p/control/control03.ctl
```

在 RMAN 中查看一下关于 controlfile 的备份：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ rman target / catalog u_rcadm/admin@catdb
Recovery Manager: Release 11.2.0.4.0 - Production on Tue May 10 14:52:31 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
```

```
connected to target database: ORA11G (DBID=12479381)
connected to recovery catalog database
RMAN> list backup of controlfile;
List of Backup Sets
===================
BS Key  Type LV  Size       Device Type  Elapsed Time  Completion Time
------  -------  ---------- -----------  ------------  ---------------
155     Full     9.58M      DISK         00:00:01      10-MAY-16
        BP Key: 164   Status: AVAILABLE  Compressed: NO  Tag: TAG20160510T110706
        Piece Name: /u01/oracle/fra/ORA11G/backupset/
2016_05_10/o1_mf_ncsnf_TAG20160510T110706_cm2n3cq4_.bkp
  Control File Included: Ckp SCN: 265816      Ckp time: 10-MAY-16
```

注意，这里需要保证 ora11g 和 catdb 两个数据库都处于 open 状态，默认监听已经启动，并且这两个数据库的服务都已注册到该监听上。

既然已经有了备份，接下来删除一个控制文件：

```
SYS@ora11g> !rm /home/oracle/backup/control/control03.ctl
```

然后手工执行检查点：

```
SYS@ora11g> alter system checkpoint;
System altered.
```

这里没有报错，我们关闭数据库：

```
SYS@ora11g> shutdown immediate;
Database closed.
ORA-00210: cannot open the specified control file
ORA-00202: control file: '/home/oracle/backup/control/control03.ctl'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

这里报错了，显示当前有一个控制文件不存在。此时，正常关闭就不行了，我们使用 abort 方式关闭：

```
SYS@ora11g> shutdown abort;
ORACLE instance shut down.
```

这里出现的问题非常简单：数据库有三个控制文件，然后删除了一个，其他两个还存在并且没有问题。我们只需要复制一个完好的过来就行。不过，这个复制操作必须在数据库关闭状态下执行。

```
SYS@ora11g> exit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
[oracle@rhel6 ~]$ cp /u01/oracle/oradata/ora11g/control01.ctl
/home/oracle/backup/control/control03.ctl
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Tue May 10 15:00:14 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SYS@ora11g> startup;
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size          2257840 bytes
Variable Size       553651280 bytes
Database Buffers    276824064 bytes
Redo Buffers        2371584 bytes
Database mounted.
Database opened.
```

此时再查看控制文件，就没有问题了。

来看第二种情况，如果三个控制文件都丢失了呢？注意，继续进行之前，一定要确保控制文件已经备份：

```
SYS@ora11g> !rm /u01/oracle/oradata/ora11g/control01.ctl
SYS@ora11g> !rm /u01/oracle/fra/control2.ctl
SYS@ora11g> !rm /home/oracle/backup/control/control03.ctl
SYS@ora11g> shutdown immediate;
Database closed.
ORA-00210: cannot open the specified control file
ORA-00202: control file: '/u01/oracle/oradata/ora11g/control01.ctl'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

依然是正常方式无法关闭数据库，需要使用 abort 方式：

```
SYS@ora11g> shutdown abort;
ORACLE instance shut down.
SYS@ora11g> startup;
ORACLE instance started.
Total System Global Area  835104768 bytes
```

```
Fixed Size          2257840 bytes
Variable Size       553651280 bytes
Database Buffers    276824064 bytes
Redo Buffers        2371584 bytes
ORA-00205: error in identifying control file, check alert log for more info
```

然后重启数据库，启动到 mount 状态时报错。

这里，需要使用 RMAN 的备份进行恢复：

```
RMAN> exit
RMAN-06900: WARNING: unable to generate V$RMAN_STATUS or V$RMAN_OUTPUT row
RMAN-06901: WARNING: disabling update of the V$RMAN_STATUS and V$RMAN_OUTPUT
rows
ORACLE error from target database:
ORA-03135: connection lost contact
Process ID: 3927
Session ID: 34 Serial number: 15
Recovery Manager complete.
[oracle@rhel6 ~]$ rman target / catalog u_rcadm/admin@catdb
Recovery Manager: Release 11.2.0.4.0 - Production on Tue May 10 15:05:43 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
connected to target database: ORA11G (not mounted)
connected to recovery catalog database
```

因为目标数据库 ora11g 已经重启，因此需要退出 rman 重新连接。与 5.4 一节一样，我们没有配置控制文件自动备份，因此只能手工指定要使用的备份文件，该备份文件必须包含控制文件的备份：

```
RMAN> restore controlfile from '/u01/oracle/fra/ORA11G/
backupset/2016_05_10/o1_mf_ncsnf_TAG20160510T110706_cm2n3cq4_.bkp';
Starting restore at 10-MAY-16
using channel ORA_DISK_1
channel ORA_DISK_1: restoring control file
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
output file name=/u01/oracle/oradata/ora11g/control01.ctl
output file name=/u01/oracle/fra/control2.ctl
output file name=/home/oracle/backup/control/control03.ctl
Finished restore at 10-MAY-16
```

注意上述命令中的阴影着重显示部分，oracle 默认生成的备份会根据你进行操作时的日期生成相应的目录。因此各位读者在按照本书进行操作时，需要找到对应的目录。当然，最好还是你直接去闪回区中，确认备份所在的目录。

然后将数据库启动到 mount 状态并恢复数据库：

```
RMAN> alter database mount;
database mounted
released channel: ORA_DISK_1
RMAN> recover database;
Starting recover at 10-MAY-16
Starting implicit crosscheck backup at 10-MAY-16
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=20 device type=DISK
Crosschecked 3 objects
Finished implicit crosscheck backup at 10-MAY-16
Starting implicit crosscheck copy at 10-MAY-16
using channel ORA_DISK_1
Crosschecked 4 objects
Finished implicit crosscheck copy at 10-MAY-16
searching for all files in the recovery area
cataloging files...
cataloging done
List of Cataloged Files
=======================
File Name: /u01/oracle/fra/ORA11G/backupset/
2016_05_10/o1_mf_ncsnf_TAG20160510T110706_cm2n3cq4_.bkp
File Name: /u01/oracle/fra/ORA11G/backupset/
2016_05_10/o1_mf_annnn_TAG20160510T110725_cm2n3fmw_.bkp
File Name: /u01/oracle/fra/ORA11G/archivelog/
2016_05_10/o1_mf_1_32_cm2n3dsf_.arc
File Name: /u01/oracle/fra/ORA11G/archivelog/
2016_05_10/o1_mf_1_34_cm316hhr_.arc
File Name: /u01/oracle/fra/ORA11G/archivelog/
2016_05_10/o1_mf_1_33_cm2nqrd8_.arc
using channel ORA_DISK_1
starting media recovery
archived log for thread 1 with sequence 32 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_10/o1_mf_1_32_cm2n3dsf_.arc
archived log for thread 1 with sequence 33 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_10/o1_mf_1_33_cm2nqrd8_.arc
archived log for thread 1 with sequence 34 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_10/o1_mf_1_34_cm316hhr_.arc
archived log for thread 1 with sequence 35 is already on disk as file
/u01/oracle/fra/redo06b.log
```

```
    archived log file name=/u01/oracle/fra/ORA11G/archivelog/
2016_05_10/o1_mf_1_32_cm2n3dsf_.arc thread=1 sequence=32
    archived log file name=/u01/oracle/fra/ORA11G/archivelog/
2016_05_10/o1_mf_1_33_cm2nqrd8_.arc thread=1 sequence=33
    archived log file name=/u01/oracle/fra/ORA11G/archivelog/
2016_05_10/o1_mf_1_34_cm316hhr_.arc thread=1 sequence=34
    archived log file name=/u01/oracle/fra/redo06b.log thread=1 sequence=35
    media recovery complete, elapsed time: 00:00:00
    Finished recover at 10-MAY-16
```

待数据库恢复完成，打开数据库：

```
RMAN> alter database open resetlogs;
database opened
new incarnation of database registered in recovery catalog
starting full resync of recovery catalog
full resync complete
```

注意，这里使用 resetlogs 方式打开数据库。因为我们使用的是备份的控制文件，因此数据库中的 redo 日志和控制文件中记录的信息已经不一致了，我们需要重置 redo 日志。此时，数据库已经处于 open 状态，我们来查看一下 redo 日志的情况：

```
SYS@ora11g> select group#,sequence#,status from v$log;
    GROUP#  SEQUENCE#  STATUS
---------- ---------- ----------------
         4          1  CURRENT
         5          0  UNUSED
         6          0  UNUSED
```

可见，原有的 redo 日志组已经全部被清空，并且 sequence 重新从 1 开始计数。建议此时对数据库做一次完全备份。当然，要先打开控制文件的自动备份：

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP on;
new RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP ON;
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete
RMAN> backup database plus archivelog;
```

5.6　数据文件丢失实验

前面，我们已经完成了参数文件和控制文件丢失的实验。接下来，如果是数据文件丢失呢？实际上，数据文件可以进行一个简单分类：关键性数据文件和非关键性文件。所谓关键性数据文件，指的是 system 表空间和 undo 表空间中的数据文件。其他数据文件则称为非关键性数据文件。之所以做这样的分类，原因也非常简单：当关键性数据文件丢失时，需要重启数据库到 mount 状态来进行恢复，非关键性数据文件则不需要。

我们先来分析关键性数据文件丢失：

```
SYS@ora11g> select file_name from dba_data_files;
FILE_NAME
--------------------------------------------------------------------------
/u01/oracle/oradata/ora11g/tbs_test_01.dbf
/u01/oracle/oradata/ora11g/tbs_test_02.dbf
/u01/oracle/oradata/ora11g/system01.dbf
/u01/oracle/oradata/ora11g/sysaux01.dbf
/u01/oracle/oradata/ora11g/undotbs01.dbf
/u01/oracle/oradata/ora11g/users01.dbf
6 rows selected.
```

这里显示了当前数据库所有的数据文件，我们将 system 表空间的数据文件删除，也就是 1 号数据文件：

```
SYS@ora11g> !rm /u01/oracle/oradata/ora11g/system01.dbf
```

然后关闭数据库：

```
SYS@ora11g> shutdown immediate;
ORA-01116: error in opening database file 1
ORA-01110: data file 1: '/u01/oracle/oradata/ora11g/system01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
SYS@ora11g> shutdown abort;
ORACLE instance shut down.
```

将数据库重新启动到 mount 状态：

```
SYS@ora11g> startup mount;
ORACLE instance started.
Total System Global Area  835104768 bytes
```

```
Fixed Size          2257840 bytes
Variable Size       553651280 bytes
Database Buffers    276824064 bytes
Redo Buffers        2371584 bytes
Database mounted.
```

然后，我们在 rman 中恢复该文件：

```
RMAN> restore datafile 1;
Starting restore at 10-MAY-16
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=21 device type=DISK
channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_DISK_1: restoring datafile 00001 to
/u01/oracle/oradata/ora11g/system01.dbf
channel ORA_DISK_1: reading from backup piece
/u01/oracle/fra/ORA11G/backupset/2016_05_10/o1_mf_nnndf_TAG20160510T151937_cm32
wb2c_.bkp
channel ORA_DISK_1: piece
handle=/u01/oracle/fra/ORA11G/backupset/2016_05_10/o1_mf_nnndf_TAG20160510T1519
37_cm32wb2c_.bkp tag=TAG20160510T151937
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
Finished restore at 10-MAY-16
RMAN> recover datafile 1;
Starting recover at 10-MAY-16
using channel ORA_DISK_1
starting media recovery
media recovery complete, elapsed time: 00:00:00
Finished recover at 10-MAY-16
RMAN> alter database open;
database opened
```

这样就恢复好了 1 号数据文件。但如果非关键性数据文件丢失呢？例如，我们将 users 表空间的数据文件删除：

```
SYS@ora11g> !rm /u01/oracle/oradata/ora11g/users01.dbf
```

该文件为 4 号数据文件，并且 users 表空间为默认系统表空间，一旦该表空间中的文件删除，就无法在此表空间中创建对象了：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create table tab_t1 as select * from emp;
create table tab_t1 as select * from emp
                               *
ERROR at line 1:
ORA-01116: error in opening database file 4
ORA-01110: data file 4: '/u01/oracle/oradata/ora11g/users01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

下面来处理这个问题。

该文件已经丢失，所以先将其置于离线状态：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> alter database datafile 4 offline;
Database altered.
```

然后在 RMAN 中恢复此文件：

```
RMAN> restore datafile 4;
Starting restore at 10-MAY-16
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_DISK_1: restoring datafile 00004 to
/u01/oracle/oradata/ora11g/users01.dbf
channel ORA_DISK_1: reading from backup piece
/u01/oracle/fra/ORA11G/backupset/2016_05_10/o1_mf_nnndf_TAG20160510T151937_cm32
wb2c_.bkp
channel ORA_DISK_1: piece
handle=/u01/oracle/fra/ORA11G/backupset/2016_05_10/o1_mf_nnndf_TAG20160510T1519
37_cm32wb2c_.bkp tag=TAG20160510T151937
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:07
Finished restore at 10-MAY-16
RMAN> recover datafile 4;
Starting recover at 10-MAY-16
using channel ORA_DISK_1
starting media recovery
```

```
media recovery complete, elapsed time: 00:00:00
Finished recover at 10-MAY-16
```

恢复完毕，则可将 4 号文件重新修改为在线状态了：

```
SYS@ora11g> alter database datafile 4 online;
Database altered.
```

然后尝试创建一个测试表：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create table tab_t1 as select * from emp;
Table created.
```

此时，就没有问题了。

非关键性文件丢失，可在数据库处于 open 状态下恢复，这样做的好处是使数据库系统具有更高的可用性。毕竟，只是一个或者几个文件丢失，其他文件仍能提供给用户使用，仅部分文件无法访问而已。如果是整个数据库无法访问，那对业务的影响就很大了。因此，在实际工作中，DBA 处理问题时，需要注意的一点是：如果能够在数据库处于 open 状态下解决问题，就尽量不要关闭数据库。这样，可以保证大部分业务不会中断，从而降低对业务的影响。对于 7*24 小时运行的系统，这一点尤为重要。

5.7 临时文件丢失实验

临时表空间中的临时文件可用来存储临时表的数据，或者当数据进行内存排序时，内存如果存储不下，临时文件也可用来存储这些内存排序的中间结果。当然这里关注的是，如果临时文件丢失了，该如何处理。

先查看当前数据库的临时文件：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> col FILE_NAME for a50
SYS@ora11g> select file_id,file_name from dba_temp_files order by 1;
   FILE_ID FILE_NAME
---------- --------------------------------------------------
         1 /u01/oracle/oradata/ora11g/temp01.dbf
         2 /u01/oracle/oradata/ora11g/temp001.dbf
         3 /home/oracle/temp002.dbf
```

前面的 4.4 一节将当前数据库的默认临时表空间设置成临时表空间组 TEMP_GRP。我们先

将数据库默认的临时表空间调整回来，再做本次实验：

```
SYS@ora11g> select tablespace_name from dba_tablespaces;
TABLESPACE_NAME
------------------------------
SYSTEM
SYSAUX
UNDOTBS1
TEMPTS1
USERS
TEMP001
TEMP002
TBS_TEST
8 rows selected.
SYS@ora11g> alter database default temporary tablespace TEMPTS1;
Database altered.
```

然后，将默认临时表空间中的临时文件删除：

```
SYS@ora11g> !rm /u01/oracle/oradata/ora11g/temp01.dbf
```

我们来创建一个临时表：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create global temporary table temp_emp1 on commit preserve rows
as select * from emp;
create global temporary table temp_emp1 on commit preserve rows as select * from
emp
                                                                          *
ERROR at line 1:
ORA-01116: error in opening database file 201
ORA-01110: data file 201: '/u01/oracle/oradata/ora11g/temp01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

数据文件有关键性和非关键性之分，但临时文件就没有了。我们可以直接添加新的临时文件，然后将已删除的临时文件从数据库的数据字典中删除：

```
SYS@ora11g> alter tablespace TEMPTS1 add tempfile
'/u01/oracle/oradata/ora11g/temp_new.dbf' size 100m;
Tablespace altered.
```

```
SYS@ora11g> alter tablespace TEMPTS1 drop tempfile
'/u01/oracle/oradata/ora11g/temp01.dbf';
Tablespace altered.
```

这样，再创建临时表就可以了：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create global temporary table temp_emp1 on commit preserve rows
as select * from emp;
Table created.
```

实际上，在 Oracle 11*g* 版本中，还有一种方法可用来解决临时文件丢失的方式，那就是重启数据库。一旦有临时文件丢失，Oracle 会在重启数据库时，自动创建临时文件。但是对于生产系统而言，这种方法需要进行数据库重启，因此会导致业务中断，因此虽然 Oracle 数据库有这样的特性，但不建议在生产系统中使用。

5.8 Oracle 11*g* 中的自动修复实验

在前面的实验中，我们虽然使用 RMAN 先后恢复了参数文件(spfile)、控制文件和数据文件(包含关键性数据文件和非关键性数据文件)，但这些恢复方法都是我们自己手动处理的。那么，Oracle 能否提供一定的自动化机制来进行这样的恢复呢？

从 11*g* 开始，Oracle 引入了修复指导的概念，或者称为自动修复。也就是说，某些条件下，Oracle 能够自动检测并发现错误，然后提供尝试修复该错误的脚本，并在 DBA 的确认之后，利用该脚本来自动修复前面检测到的错误。

实际上，自动化的概念是 Oracle 从 9*i* 版本进化到 10*g* 版本时最重要的特性。从 10*g* 版本开始，Oracle 能在一定程度上完成数据库的自动检测与管理以及 SQL 辅助优化等工作。也就是说，Oracle 期望能将相当一部分 DBA 工作通过自动化管理来完成，而不再依赖于 DBA 的手工操作。对于 DBA，尤其对于新手 DBA 来说，这样的事情显然是很好的。因为它降低了 DBA 入门的难度，使得不少问题的处理能够更简捷。从 11*g* 开始，Oracle 又在数据库故障处理方面，做了进一步自动化处理的尝试。

我们先删除一个文件，然后看 Oracle 如何提供修复指导来帮助 DBA 解决问题。

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> !rm /u01/oracle/oradata/ora11g/users01.dbf
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create table tab_t1 as select * from emp;
```

```
create table tab_t1 as select * from emp
                              *
ERROR at line 1:
ORA-01116: error in opening database file 4
ORA-01110: data file 4: '/u01/oracle/oradata/ora11g/users01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

这里依然删除 4 号数据文件，然后进入 RMAN：

```
RMAN> list failure;
starting full resync of recovery catalog
full resync complete
List of Database Failures
=========================
Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
642        HIGH     OPEN      10-MAY-16     One or more non-system datafiles are
missing
```

执行 list failure 命令，结果显示 Oracle 检测到了错误，它提示说当前数据库有一个或者多个非系统性文件丢失。显然，我们前面文件删除的故障，Oracle 检测到了。

接下来，我们继续：

```
RMAN> advise failure;
List of Database Failures
=========================
Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
642        HIGH     OPEN      10-MAY-16     One or more non-system datafiles are
missing

analyzing automatic repair options; this may take some time
using channel ORA_DISK_1
analyzing automatic repair options complete
Mandatory Manual Actions
========================
no manual actions available

Optional Manual Actions
=======================
```

```
    1. If file /u01/oracle/oradata/ora11g/users01.dbf was unintentionally renamed
or moved, restore it

    Automated Repair Options
    ========================
    Option Repair Description
    ------ ------------------
    1      Restore and recover datafile 4
      Strategy: The repair includes complete media recovery with no data loss
      Repair script: /u01/oracle/diag/rdbms/ora11g/ora11g/hm/reco_3082837339.hm
```

我们执行 advise failure 命令。也就是说，既然 Oracle 检测到了故障，那能否给点建议呢？该命令的输出分为三个部分：第一部分为检测到的故障，第二部分为手工处理选项，第三个为自动修复选项。我们着重分析第三部分。这里，advise failure 命令的自动修复选项提供了一个修复的脚本，我们来看该脚本的内容：

```
    [oracle@rhel6 ~]$ more
/u01/oracle/diag/rdbms/ora11g/ora11g/hm/reco_3082837339.hm
       # restore and recover datafile
       sql 'alter database datafile 4 offline';
       restore datafile 4;
       recover datafile 4;
       sql 'alter database datafile 4 online';
```

这显然和我们前面处理的命令完全一样！可见，Oracle 的自动修复还是管用的。

接下来执行修复：

```
RMAN> repair failure;
Strategy: The repair includes complete media recovery with no data loss
Repair script: /u01/oracle/diag/rdbms/ora11g/ora11g/hm/reco_3082837339.hm
contents of repair script:
   # restore and recover datafile
   sql 'alter database datafile 4 offline';
   restore datafile 4;
   recover datafile 4;
   sql 'alter database datafile 4 online';

Do you really want to execute the above repair (enter YES or NO)?
```

这里提示你是否确认要执行该修复脚本，我们输入 yes。

```
executing repair script
```

```
    sql statement: alter database datafile 4 offline
    Starting restore at 10-MAY-16
    using channel ORA_DISK_1
    channel ORA_DISK_1: starting datafile backup set restore
    channel ORA_DISK_1: specifying datafile(s) to restore from backup set
    channel ORA_DISK_1: restoring datafile 00004 to
/u01/oracle/oradata/ora11g/users01.dbf
    channel ORA_DISK_1: reading from backup piece
/u01/oracle/fra/ORA11G/backupset/2016_05_10/o1_mf_nnndf_TAG20160510T151937_cm32
wb2c_.bkp
    channel ORA_DISK_1: piece
handle=/u01/oracle/fra/ORA11G/backupset/2016_05_10/o1_mf_nnndf_TAG20160510T1519
37_cm32wb2c_.bkp tag=TAG20160510T151937
    channel ORA_DISK_1: restored backup piece 1
    channel ORA_DISK_1: restore complete, elapsed time: 00:00:07
    Finished restore at 10-MAY-16
    Starting recover at 10-MAY-16
    using channel ORA_DISK_1
    starting media recovery
    media recovery complete, elapsed time: 00:00:01
    Finished recover at 10-MAY-16
    sql statement: alter database datafile 4 online
    repair failure complete
```

自动修复完成，我们去查看一下：

```
[oracle@rhel6 ~]$ cd /u01/oracle/oradata/ora11g/
[oracle@rhel6 ora11g]$ ls
control01.ctl  redo06a.log   tbs_test_01.dbf temp_new.dbf
redo04a.log    sysaux01.dbf  tbs_test_02.dbf undotbs01.dbf
redo05a.log    system01.dbf  temp001.dbf     users01.dbf
```

可见，4 号数据文件确实已经恢复回来了。

通过一个简单的测试可知，Oracle 11g 提供的这个自动修复是非常有效的。但这个特性的使用，需要满足如下两个条件：

1) 数据库拥有完整的 RMAN 备份

2) 数据库实例已经被启动到 mount 状态

当然，即便是满足这些条件，也不意味着该自动修复功能真能处理所有的数据库故障。也就是说，很多时候，还是依赖于 DBA 的水平，你需要自己来处理问题，而不是仅指望自动修复。

另外，使用该特性时，list failure、advise failure 和 repair failure 三条命令的执行顺序不能颠倒，必须严格按照 list、advise 和 repair 的顺序执行。并且，只有在 list failure 确实检测到了数据库故障，才能执行 advise failure 命令。同样，只有 advise failure 确实能够提供自动修复脚本，才能执行 repair failure 命令。也就是说，这三条命令之间存在严格的依赖关系。

5.9 redo 文件损坏恢复实验

关于 redo 的内容，我们在前面 4.2 一节中已对当前数据库的 redo 日志进行了调整，一旦 redo 日志组创建并开始使用后，redo 日志组就会有三种状态：current、active 和 inactive。那么，如果处于不同状态的 redo 日志文件丢失或者损坏，该如何恢复？

先看 inactive 状态的日志组：

```
SYS@ora11g> select group#,sequence#,status,archived from v$log;
    GROUP#  SEQUENCE# STATUS           ARC
---------- ---------- ---------------- ---
         4          4 CURRENT          NO
         5          2 INACTIVE         YES
         6          3 INACTIVE         YES
SYS@ora11g> select member from v$logfile;
MEMBER
--------------------------------------------------------------------------
/u01/oracle/oradata/ora11g/redo04a.log
/u01/oracle/fra/redo04b.log
/u01/oracle/oradata/ora11g/redo05a.log
/u01/oracle/fra/redo05b.log
/u01/oracle/oradata/ora11g/redo06a.log
/u01/oracle/fra/redo06b.log

6 rows selected.
```

此时第 5 和第 6 两个 redo 日志组都处于 inactive 状态，我们将第 6 组中的一个 redo 日志文件删除：

```
SYS@ora11g> !rm /u01/oracle/oradata/ora11g/redo06a.log
```

然后进行日志切换：

```
SYS@ora11g> alter system switch logfile;
System altered.
SYS@ora11g> /
```

```
System altered.
SYS@ora11g> /
SYS@ora11g> /
System altered.
SYS@ora11g> /
System altered.
```

可以看到，尽管丢了一个日志文件，但是 redo 日志组的切换毫无问题。此时查看 alert 日志，可以看到：

```
[oracle@rhel6 ~]$ tail -f
/u01/oracle/diag/rdbms/ora11g/ora11g/trace/alert_ora11g.log
Checkpoint not complete
  Current log# 6 seq# 6 mem# 0: /u01/oracle/oradata/ora11g/redo06a.log
  Current log# 6 seq# 6 mem# 1: /u01/oracle/fra/redo06b.log
Tue May 10 16:51:38 2016
Thread 1 advanced to log sequence 7 (LGWR switch)
  Current log# 4 seq# 7 mem# 0: /u01/oracle/oradata/ora11g/redo04a.log
  Current log# 4 seq# 7 mem# 1: /u01/oracle/fra/redo04b.log
Tue May 10 16:51:38 2016
Errors in file /u01/oracle/diag/rdbms/ora11g/ora11g/trace/ora11g_arc3_5040.trc:
ORA-00313: open failed for members of log group 6 of thread 1
ORA-00312: online log 6 thread 1: '/u01/oracle/oradata/ora11g/redo06a.log'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
Errors in file
/u01/oracle/diag/rdbms/ora11g/ora11g/trace/ora11g_arc3_5040.trc:
ORA-00313: open failed for members of log group 6 of thread 1
ORA-00312: online log 6 thread 1: '/u01/oracle/oradata/ora11g/redo06a.log'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
Deleted Oracle managed file
/u01/oracle/fra/ORA11G/archivelog/2016_05_10/o1_mf_1_34_cm316hhr_.arc
Archived Log entry 25 added for thread 1 sequence 6 ID 0xc2a452 dest 1:
Tue May 10 16:51:38 2016
Errors in file
/u01/oracle/diag/rdbms/ora11g/ora11g/trace/ora11g_m000_6763.trc:
ORA-00313: open failed for members of log group 6 of thread 1
ORA-00312: online log 6 thread 1: '/u01/oracle/oradata/ora11g/redo06a.log'
```

```
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
Checker run found 1 new persistent data failures
Thread 1 advanced to log sequence 8 (LGWR switch)
  Current log# 5 seq# 8 mem# 0: /u01/oracle/oradata/ora11g/redo05a.log
  Current log# 5 seq# 8 mem# 1: /u01/oracle/fra/redo05b.log
Tue May 10 16:51:39 2016
Deleted Oracle managed file
/u01/oracle/fra/ORA11G/archivelog/2016_05_10/o1_mf_1_33_cm2nqrd8_.arc
Archived Log entry 26 added for thread 1 sequence 7 ID 0xc2a452 dest 1:
```

可以看到，告警日志中实际上已经检测到有 redo 日志文件丢失，但是整个数据库依然能够正常运行。这就显示出 redo 日志多路复用的价值了。也就是说，只要一个 redo 日志组中还有一个 redo 日志，数据库就能始终保持正常运行状态。

那如果把第 6 组中剩下的那个 redo 日志也删除呢？

```
SYS@ora11g> select group#,sequence#,status,archived from v$log;
   GROUP#  SEQUENCE# STATUS           ARC
---------- ---------- ---------------- ---
  4     7 INACTIVE       YES
  5     8 CURRENT        NO
  6     6 INACTIVE       YES
```

注意，这里仍然保持第 6 组 redo 日志组处于 inactive 状态。

```
SYS@ora11g> !rm /u01/oracle/fra/redo06b.log
SYS@ora11g> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SYS@ora11g>  startup
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size      2257840 bytes
Variable Size    553651280 bytes
Database Buffers  276824064 bytes
Redo Buffers      2371584 bytes
Database mounted.
ORA-03113: end-of-file on communication channel
Process ID: 6987
```

```
Session ID: 1 Serial number: 5
```

在重新启动数据库的过程中，报告 3113 错误。我们来分析该错误信息：

```
[oracle@rhel6 ~]$ oerr ora 3113
03113, 00000, "end-of-file on communication channel"
// *Cause: The connection between Client and Server process was broken.
// *Action: There was a communication error that requires further investigation.
//         First, check for network problems and review the SQL*Net setup.
//         Also, look in the alert.log file for any errors. Finally, test to
//         see whether the server process is dead and whether a trace file
//         was generated at failure time.
```

3113 错误意指数据库实例停止运行了，因此你和数据库的连接断开了：

```
SYS@ora11g> exit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Tue May 10 17:02:53 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SYS@ora11g>
```

可见，此时我们连接到的是一个 idle 实例，也就是空实例。这表示数据库实例已经崩溃了。

为处理该问题，我们需要先将数据库启动到 mount 状态，因为当前数据库的参数文件和控制文件没有问题，因此可以启动到 mount 状态：

```
SYS@ora11g> startup mount;
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size          2257840 bytes
Variable Size       553651280 bytes
Database Buffers    276824064 bytes
Redo Buffers        2371584 bytes
Database mounted.
```

然后将 redo 文件丢失的日志组进行清理，并打开数据库：

```
SYS@ora11g> alter database clear logfile group 6;
Database altered.
SYS@ora11g> alter database open;
```

```
Database altered.
```

此时，再来查看 redo 日志组的状态：

```
SYS@ora11g> select group#,sequence#,status,archived from v$log;
    GROUP#  SEQUENCE#  STATUS           ARC
---------- ---------- ---------------- ---
         4          7 INACTIVE         YES
         5          8 CURRENT          NO
         6          0 UNUSED           YES
```

可见，第 6 组 redo 日志已被重置。前面的 clear 命令实际上就是将 redo 日志组清空重新使用。

那如果处于 active 状态的 redo 日志组丢失呢？

```
SYS@ora11g> alter system switch logfile;
System altered.
SYS@ora11g> select group#,sequence#,status,archived from v$log;

    GROUP#  SEQUENCE#  STATUS            ARC
---------- ---------- -------------  ----
         4          7 INACTIVE         YES
         5          8 ACTIVE           YES
         6          9 CURRENT          NO
SYS@ora11g> !rm /u01/oracle/oradata/ora11g/redo05a.log
SYS@ora11g> !rm /u01/oracle/fra/redo05b.log
```

我们重启数据库：

```
SYS@ora11g> shutdown abort;
ORACLE instance shut down.
SYS@ora11g> startup
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size      2257840 bytes
Variable Size    553651280 bytes
Database Buffers  276824064 bytes
Redo Buffers      2371584 bytes
Database mounted.
ORA-03113: end-of-file on communication channel
Process ID: 7401
Session ID: 1 Serial number: 5
```

当然，数据库实例又崩溃了。

```
SYS@ora11g> exit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Tue May 10 17:11:17 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SYS@ora11g> startup mount;
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size          2257840 bytes
Variable Size       553651280 bytes
Database Buffers    276824064 bytes
Redo Buffers        2371584 bytes
Database mounted.
```

我们尝试清理该日志组：

```
SYS@ora11g> alter database clear logfile group 5;
Database altered.
SYS@ora11g> alter database open;
Database altered.
```

再查看 redo 日志组的状态：

```
SYS@ora11g> select group#,sequence#,status,archived from v$log;
    GROUP#  SEQUENCE#  STATUS          ARC
---------- ---------- ------------  ---
         4          7  INACTIVE        YES
         5         10  CURRENT         NO
         6          9  INACTIVE        YES
```

最后，我们来看处于 current 状态的 redo 日志组出现问题时的情形。

```
SYS@ora11g> alter system switch logfile;
System altered.
SYS@ora11g>  select group#,sequence#,status,archived from v$log;
    GROUP#  SEQUENCE#  STATUS           ARC
---------- ---------- -------------  ---
         4         11  INACTIVE         YES
```

```
         5          13 CURRENT       NO
         6          12 ACTIVE        YES
```

当前处于 current 状态的日志组为第 5 组。这里，不再像前面那样直接删除该组的 redo 文件，而使用其他文件来覆盖 redo 文件的内容：

```
SYS@ora11g> !cp /etc/passwd /u01/oracle/oradata/ora11g/redo05a.log
SYS@ora11g> !cp /etc/passwd /u01/oracle/fra/redo05b.log
```

这里用到的/etc/passwd 文件是 Linux 系统中用来存储所有用户账户信息的地方。然后做一次日志切换：

```
SYS@ora11g> alter system switch logfile;
alter system switch logfile
*
ERROR at line 1:
ORA-03113: end-of-file on communication channel
Process ID: 3762
Session ID: 1 Serial number: 5
```

又报出 3113 的错误，显然，数据库实例又崩溃了。

重新连接到 sqlplus，然后将数据库启动到 mount 状态：

```
SYS@ora11g> exit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Wed May 11 09:20:11 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SYS@ora11g> startup mount;
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size          2257840 bytes
Variable Size       553651280 bytes
Database Buffers  276824064 bytes
Redo Buffers        2371584 bytes
Database mounted.
SYS@ora11g>
```

然后，按照前面的做法，尝试清除(clear)第 5 组 redo：

```
SYS@ora11g> alter database clear logfile group 5;
alter database clear logfile group 5
*
ERROR at line 1:
ORA-01624: log 5 needed for crash recovery of instance ora11g (thread 1)
ORA-00312: online log 5 thread 1: '/u01/oracle/oradata/ora11g/redo05a.log'
ORA-00312: online log 5 thread 1: '/u01/oracle/fra/redo05b.log'
```

那我们加上 unarchived，再试试：

```
SYS@ora11g> alter database clear unarchived logfile group 5;
alter database clear unarchived logfile group 5
*
ERROR at line 1:
ORA-01624: log 5 needed for crash recovery of instance ora11g (thread 1)
ORA-00312: online log 5 thread 1: '/u01/oracle/oradata/ora11g/redo05a.log'
ORA-00312: online log 5 thread 1: '/u01/oracle/fra/redo05b.log'
```

还是不行。前面 5.8 一节提到自动修复，那就在 RMAN 中试试：

```
[oracle@rhel6 ~]$ rman target sys/oracle@ora11g catalog u_rcadm/admin@catdb
Recovery Manager: Release 11.2.0.4.0 - Production on Wed May 11 09:25:55 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
connected to target database: ORA11G (DBID=12479381, not open)
connected to recovery catalog database
RMAN> list failure;
List of Database Failures
=========================
Failure ID Priority  Status  Time Detected  Summary
---------- --------  ------  -------------  -------
873        CRITICAL  OPEN    10-MAY-16      Redo log group 5 is unavailable
893        HIGH      OPEN    11-MAY-16      Redo log file /u01/oracle/fra/
redo05b.log is corrupt
870        HIGH      OPEN    10-MAY-16      Redo log file /u01/oracle/
oradata/ora11g/redo05a.log is corrupt
```

对于 redo 文件损坏的情况，oracle 监测到了。oracle 提出什么建议呢？

```
RMAN> advise failure;
List of Database Failures
=========================
Failure ID Priority Status  Time Detected Summary
---------- -------- ------- ------------- -------
```

```
    873        CRITICAL  OPEN    10-MAY-16     Redo log group 5 is unavailable
    893        HIGH      OPEN    11-MAY-16     Redo log file /u01/oracle/fra/
redo05b.log is corrupt
    870        HIGH      OPEN    10-MAY-16     Redo log file /u01/oracle/oradata/
ora11g/redo05a.log is corrupt

    analyzing automatic repair options; this may take some time
    allocated channel: ORA_DISK_1
    channel ORA_DISK_1: SID=21 device type=DISK
    analyzing automatic repair options complete

    Mandatory Manual Actions
    ========================
    no manual actions available

    Optional Manual Actions
    =======================
    no manual actions available

    Automated Repair Options
    ========================
    Option Repair Description
    ------ ------------------
    1      Perform flashback of the database to SCN 371853
      Strategy: The repair includes flashing the database back with some data loss
      Repair script: /u01/oracle/diag/rdbms/ora11g/ora11g/hm/reco_3010304523.hm
```

这里，oracle 提供了自动修复的脚本，我们来分析该脚本的内容：

```
    [root@rhel6 Desktop]# su - oracle
    [oracle@rhel6 ~]$ more /u01/oracle/diag/rdbms/ora11g/ora11g/hm/
reco_3010304523.hm
       # flashback database
       flashback database to before scn 371853;
       alter database open resetlogs;
```

显然，oracle 利用了闪回技术处理这样的故障。我们继续：

```
RMAN> repair failure;
Strategy: The repair includes flashing the database back with some data loss
Repair script: /u01/oracle/diag/rdbms/ora11g/ora11g/hm/reco_3010304523.hm
contents of repair script:
```

```
  # flashback database
  flashback database to before scn 371853;
  alter database open resetlogs;
Do you really want to execute the above repair (enter YES or NO)?
```

我们输入 yes，然后回车：

```
executing repair script
Starting flashback at 11-MAY-16
using channel ORA_DISK_1
starting media recovery
archived log for thread 1 with sequence 4 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_10/o1_mf_1_4_cm388gl8_.arc
archived log for thread 1 with sequence 5 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_10/o1_mf_1_5_cm388j0y_.arc
archived log for thread 1 with sequence 6 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_10/o1_mf_1_6_cm388t6y_.arc
archived log for thread 1 with sequence 7 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_10/o1_mf_1_7_cm388vqn_.arc
archived log for thread 1 with sequence 8 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_10/o1_mf_1_8_cm39987v_.arc
archived log for thread 1 with sequence 9 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_10/o1_mf_1_9_cm39hpmk_.arc
archived log for thread 1 with sequence 10 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_10/o1_mf_1_10_cm39ooff_.arc
archived log for thread 1 with sequence 11 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_11/o1_mf_1_11_cm51o8t6_.arc
archived log for thread 1 with sequence 12 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_11/o1_mf_1_12_cm51whot_.arc
media recovery complete, elapsed time: 00:00:08
Finished flashback at 11-MAY-16

database opened
repair failure complete
new incarnation of database registered in recovery catalog
starting full resync of recovery catalog
full resync complete
```

数据库已经以 resetlogs 方式打开，我们看一下此时 redo 日志组的情况：

```
SYS@ora11g> select group#,sequence#,status,archived from v$log;
   GROUP# SEQUENCE#  STATUS      ARC
```

```
---------- ---------- ----------- ---
         4          1 CURRENT     NO
         5          0 UNUSED      YES
         6          0 UNUSED      YES
```

此时，所有 redo 日志组都被清除，重新开始，sequence 也从 1 开始重新计数。完成故障处理后，跟前面一样，记得立即对数据库做一个完全备份。当然，鉴于 redo 已经重新开始计数，前面的备份已经出现断层，因此，我们可以把以前的备份都删除，然后再进行完全备份：

```
RMAN> delete noprompt backup;
```

输出省略。

```
RMAN> backup database plus archivelog;
```

至此，我们把处于不同状态的redo日志组的丢失问题都处理完了。实际上，当current状态的日志组出现问题时，我们还有一种方法来处理，那就是直接跳过损坏的redo日志组强制开启数据库。此时需要用到oracle的一个隐含参数：_allow_resetlogs_corruption。这里暂不做赘述，我们在后续书籍中再进行说明及使用。

关于闪回，我们接下来进行分析。

5.10 数据库闪回实验合集

在 Oracle 中，处理数据丢失的问题，除了可以利用备份进行恢复之外，Oracle 还提供一种极其有效且便捷的技术——闪回(flashback)。所谓闪回，实质上就是尽量利用数据库现有的技术来实现轻量级的数据恢复。它是常规备份恢复的有效补充，当然，限制也相当多。

在 11g 数据库中，闪回共分为六个部分：

- 闪回表删除
- 闪回查询
- 闪回事务查询
- 闪回版本查询
- 闪回数据库
- 闪回归档

其中，闪回表删除依赖于 recyclebin(回收站)，闪回查询、闪回事务查询和闪回版本查询依赖于 undo 表空间，闪回数据库依赖于闪回日志(当数据库开启闪回功能后，自动生成并保存在闪回区中的日志)，闪回归档则是 11g 新引入的特性，它需要单独在表空间中分配空间，用来存储某一个表或者多个表上的历史变化情况。

先说闪回表删除。

在 Oracle 数据库的表空间中，有一个逻辑区域，称为 recyclebin。当然，只局限于普通的

用户表空间，例如 users 表空间，以及其他用户自己创建的表空间。system 表空间则没有此区域。Oracle 在实际对一个表进行 drop 操作时，并没有真正将表中的数据全部删除，而将表改变了名称，并将修改后的名称存储到回收站中。这样，一旦将来某一时刻用户意识到操作错误，就可将该表从回收站中恢复回来。当然，前提该表在回收站中还存在。

数据库是否开启 recyclebin 取决于如下初始化参数：

```
SYS@ora11g> show parameter recyclebin;
NAME          TYPE        VALUE
------------- ----------- -------
recyclebin    string      on
```

默认情况下，回收站功能是开启的。

接下来，来看相关的实验。

先在 scott 用户下创建测试表。

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> select * from tab;
TNAME         TABTYPE CLUSTERID
------------- ------- ----------
BONUS         TABLE
DEPT          TABLE
EMP           TABLE
SALGRADE      TABLE
TAB_OBJ1      TABLE
TAB_T1        TABLE
TAB_TEST      TABLE
TEMP_EMP      TABLE
TEMP_EMP1     TABLE
TEMP_EMP_NEW  TABLE
V_TEST        VIEW

11 rows selected.
```

基于前面做的实验，scott 用户已有不少测试表，我们重新初始化 scott 用户：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> @?/rdbms/admin/utlsampl.sql;
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
```

```
With the Partitioning, OLAP, Data Mining and Real Application Testing options
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Wed May 11 09:55:46 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> select * from tab;
TNAME          TABTYPE  CLUSTERID
-------------- -------- -----------
BONUS          TABLE
DEPT           TABLE
EMP            TABLE
SALGRADE       TABLE
```

此时，scott 用户的 recyclebin 中是没有任何对象的：

```
SCOTT@ora11g> show recyclebin;
SCOTT@ora11g>
```

创建测试表 tab_emp：

```
SCOTT@ora11g> create table tab_emp as select * from emp;
Table created.
SCOTT@ora11g> select count(*) from tab_emp;
  COUNT(*)
----------
 14
```

然后将该表删除：

```
SCOTT@ora11g> drop table tab_emp;
Table dropped.
```

再查看回收站：

```
SCOTT@ora11g> show recyclebin;
ORIGINAL NAME  RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
-------------- ---------------                ------------ --------------
TAB_EMP        BIN$Mohh64Z7Ev7gU2/IqMBxxw==$0 TABLE      2016-05-11:09:58:48
```

注意这里的 recyclebin name，这就是 tab_emp 在回收站中的名称。同时该输出也显示了删

除的对象类型及删除时间。

如果想利用闪回表删除来恢复该表，操作命令其实非常简单：

```
SCOTT@ora11g> flashback table tab_emp to before drop;
Flashback complete.
```

此时回收站就空了：

```
SCOTT@ora11g> show recyclebin;
```

tab_emp 也恢复回来了：

```
SCOTT@ora11g> select * from tab;
TNAME          TABTYPE CLUSTERID
------------- ------------------
BONUS          TABLE
DEPT           TABLE
EMP            TABLE
SALGRADE       TABLE
TAB_EMP        TABLE
```

显然，这是一个很好用的功能。如果我们在删除表的时候，想永久删除，而不将其放在回收站里，则可以使用 purge：

```
SCOTT@ora11g> drop table tab_emp purge;
Table dropped.
SCOTT@ora11g> show recyclebin;
```

此时，回收站中就没有该表了。

闪回表删除到这里，还远没有完事。我们稍微深入一下。考虑如下情况：如果我们创建了一张表，然后删除，再创建再删除，这样 recyclebin 中就会有同名的对象存在，那么如何去确认哪个表是我们想要恢复的表？

来看如下实验：

```
SCOTT@ora11g> create table tab_emp as select * from emp;
Table created.
SCOTT@ora11g> drop table tab_emp;
Table dropped.
SCOTT@ora11g> create table tab_emp as select * from dept;
Table created.
SCOTT@ora11g> drop table tab_emp;
Table dropped.
```

此时，recyclebin 中就有两个同名对象：

```
SCOTT@ora11g> show recyclebin;
ORIGINAL NAME RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
------------- ------------------------------ ------------ ----------------
TAB_EMP       BIN$Mohh64Z9Ev7gU2/IqMBxxw==$0 TABLE        2016-05-11:10:20:35
TAB_EMP       BIN$Mohh64Z8Ev7gU2/IqMBxxw==$0 TABLE        2016-05-11:10:20:14
```

那么，如果我们想恢复其中一张表，该如何确认呢？依照上面的输出结果，也就 recyclebin name 和 drop table 两个内容不相同。但在实际环境中，如果两个表删除时间相差不远，我们很难区分究竟在哪个具体时间是删除了哪个表。因此，我们只有使用 recyclebin name 来区分对象了。当然，如果不仔细看，这两个对象的 recyclebin name 你也可能认为是一样的。

实际上，可使用该名称来查看表的结构和数据，只不过要加双引号：

```
SCOTT@ora11g> desc "BIN$Mohh64Z9Ev7gU2/IqMBxxw==$0";
 Name        Null?          Type
 ----------- -------------- --------------------
 DEPTNO      NUMBER(2)
 DNAME       VARCHAR2(14)
 LOC         VARCHAR2(13)
SCOTT@ora11g> select count(*) from "BIN$Mohh64Z9Ev7gU2/IqMBxxw==$0";
  COUNT(*)
----------
  4
```

这样，就可以确定要恢复的是哪个表了。然后我们来恢复：

```
SCOTT@ora11g> flashback table "BIN$Mohh64Z9Ev7gU2/IqMBxxw==$0" to before drop;
Flashback complete.
SCOTT@ora11g> select * from tab;
TNAME                          TABTYPE CLUSTERID
------------------------------ ------- ----------
BIN$Mohh64Z8Ev7gU2/IqMBxxw==$0 TABLE
BONUS                          TABLE
DEPT                           TABLE
EMP                            TABLE
SALGRADE                       TABLE
TAB_EMP                        TABLE

6 rows selected.
```

另外，如果恢复时，当前用户下已经有重名的对象，又当如何处理？

比如在前面的实验中我们已经恢复了 tab_emp 表，但实际上当前 recyclebin 中还有一个叫做 tab_emp 的表。如果想把这张表也恢复，又该如何处理：

```
SCOTT@ora11g> flashback table "BIN$Mohh64Z8Ev7gU2/IqMBxxw==$0" to before drop;
flashback table "BIN$Mohh64Z8Ev7gU2/IqMBxxw==$0" to before drop
*
ERROR at line 1:
ORA-38312: original name is used by an existing object
```

直接恢复是会报错的，我们需要使用 rename：

```
SCOTT@ora11g> flashback table "BIN$Mohh64Z8Ev7gU2/IqMBxxw==$0" to before drop
rename to tab_emp_new;
Flashback complete.
```

还有，如果表上有索引，则在闪回表删除时，索引又将如何处理？

我们先创建测试表及索引：

```
SCOTT@ora11g>  create table tab_obj as select * from user_objects;
Table created.
SCOTT@ora11g> create index ind_id on tab_obj(object_id);
Index created.
```

然后删除该表：

```
SCOTT@ora11g> drop table tab_obj;
Table dropped.
SCOTT@ora11g> show recyclebin;
ORIGINAL NAME RECYCLEBIN NAME                 OBJECT TYPE  DROP TIME
------------- ------------------------------ ------------ -------------------
TAB_OBJ       BIN$Mohh64Z/Ev7gU2/IqMBxxw==$0 TABLE        2016-05-11:10:34:41
```

此时，我们使用 show recyclebin 命令只能看到已经被删除的表，那索引呢？这里需要使用另一种查看 recyclebin 中对象的方法了。我们使用数据字典表 user_recyclebin：

```
SCOTT@ora11g> desc user_recyclebin;
 Name            Null?                        Type
 --------------- ------------------------------------
 OBJECT_NAME     NOT NULL VARCHAR2(30)
 ORIGINAL_NAME   VARCHAR2(32)
 OPERATION       VARCHAR2(9)
 TYPE            VARCHAR2(25)
 TS_NAME         VARCHAR2(30)
```

```
CREATETIME      VARCHAR2(19)
DROPTIME        VARCHAR2(19)
DROPSCN         NUMBER
PARTITION_NAME  VARCHAR2(32)
CAN_UNDROP      VARCHAR2(3)
CAN_PURGE       VARCHAR2(3)
RELATED         NOT NULL NUMBER
BASE_OBJECT     NOT NULL NUMBER
PURGE_OBJECT    NOT NULL NUMBER
SPACE           NUMBER
```

这样，就可以查看到已被删除的表和索引了：

```
SCOTT@ora11g> set line 200
SCOTT@ora11g> col OBJECT_NAME for a40
SCOTT@ora11g> col ORIGINAL_NAME for a15
SCOTT@ora11g> col TYPE for a10
SCOTT@ora11g> select OBJECT_NAME,ORIGINAL_NAME,TYPE,DROPTIME from
user_recyclebin;
OBJECT_NAME                    ORIGINAL_NAME TYPE   DROPTIME
------------------------------ ------------- ------ -------------------
BIN$Mohh64Z+Ev7gU2/IqMBxxw==$0  IND_ID       INDEX  2016-05-11:10:34:41
BIN$Mohh64Z/Ev7gU2/IqMBxxw==$0  TAB_OBJ      TABLE  2016-05-11:10:34:41
```

接下来，我们闪回表：

```
SCOTT@ora11g> flashback table tab_obj to before drop;
Flashback complete.
```

再查询 user_recyclebin:

```
SCOTT@ora11g> select OBJECT_NAME,ORIGINAL_NAME,TYPE,DROPTIME from
user_recyclebin;
no rows selected
```

可见，索引也被一并闪回了。但是：

```
SCOTT@ora11g> select index_name from ind;
INDEX_NAME
------------------------------
BIN$Mohh64Z+Ev7gU2/IqMBxxw==$0
PK_EMP
PK_DEPT
```

注意该索引的名称。也就是说，虽然索引被一并闪回了，但名称却没有自动更改过来。我们需要手工处理一下：

```
SCOTT@ora11g> alter index "BIN$Mohh64Z+Ev7gU2/IqMBxxw==$0" rename to ind_id;
Index altered.
```

最后一个问题，recyclebin 终究占用的是表空间里面的空间，那如果该表空间中剩余空间不足时，Oracle 是否会自动清除 recyclebin？如果该表空间中的数据文件已经打开自动扩展并且还没有扩展到最大值时，Oracle 是保留回收站中的内容然后进行自动扩展，还是先把回收站清空再自动扩展？

一切，都通过实验来回答。

scott 用户的默认表空间是 users：

```
SCOTT@ora11g> desc user_users;
 Name                      Null?    Type
 ----------------------------- ------------------------------
 USERNAME                  NOT NULL VARCHAR2(30)
 USER_ID                   NOT NULL NUMBER
 ACCOUNT_STATUS            NOT NULL VARCHAR2(32)
 LOCK_DATE                          DATE
 EXPIRY_DATE                        DATE
 DEFAULT_TABLESPACE        NOT NULL VARCHAR2(30)
 TEMPORARY_TABLESPACE      NOT NULL VARCHAR2(30)
 CREATED                   NOT NULL DATE
 INITIAL_RSRC_CONSUMER_GROUP        VARCHAR2(30)
 EXTERNAL_NAME                      VARCHAR2(4000)
SCOTT@ora11g> select USERNAME,DEFAULT_TABLESPACE from user_users;
USERNAME       DEFAULT_TABLESPACE
------------------------------------------------------------
SCOTT          USERS
```

在前面的实验中，我们知道 users 表空间只有一个数据文件，并且该数据文件的编号为 4 号。我们来查看该文件的当前大小以及是否开启自动扩展：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> select file_id,BYTES/1024/1024,AUTOEXTENSIBLE,MAXBYTES/1024/1024
from dba_data_files where file_id=4;
   FILE_ID  BYTES/1024/1024 AUT MAXBYTES/1024/1024
---------- --------------- --- ------------------
         4             500 YES         32767.9844
```

当前 4 号文件大小为 500M，自动扩展打开，文件最大可以扩展到 32GB。

为便于实验，我们将重置该文件的大小，并关闭自动扩展。

```
SYS@ora11g> alter database datafile 4 resize 100M;
Database altered.
SYS@ora11g> alter database datafile 4 autoextend off;
Database altered.
SYS@ora11g> select file_id,BYTES/1024/1024,AUTOEXTENSIBLE,MAXBYTES/1024/1024
from dba_data_files where file_id=4;
   FILE_ID  BYTES/1024/1024  AUT MAXBYTES/1024/1024
----------  ---------------  --- ------------------
         4              100  NO  0
```

然后，我们删除几张表：

```
SYS@ora11g> conn scott/tiger
Connected.
SCOTT@ora11g> select * from tab;
TNAME          TABTYPE CLUSTERID
-------------  -------------------------
BONUS          TABLE
DEPT           TABLE
EMP            TABLE
SALGRADE       TABLE
TAB_EMP        TABLE
TAB_EMP_NEW    TABLE
TAB_OBJ        TABLE

7 rows selected.
SCOTT@ora11g> drop table TAB_EMP;
Table dropped.
SCOTT@ora11g> drop table TAB_EMP_NEW;
Table dropped.
SCOTT@ora11g> drop table TAB_OBJ;
Table dropped.
SCOTT@ora11g> show recyclebin;
ORIGINAL NAME  RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
-------------  ------------------------------ ------------ ----------------
TAB_EMP        BIN$Mokib1dmGjngU2/IqMBbnQ==$0  TABLE     2016-05-11:10:52:38
TAB_EMP_NEW    BIN$Mokib1dnGjngU2/IqMBbnQ==$0  TABLE     2016-05-11:10:52:44
TAB_OBJ        BIN$Mokib1dpGjngU2/IqMBbnQ==$0  TABLE     2016-05-11:10:52:49
```

接下来，我们在 scott 用户下创建一个大表，将 users 表空间的剩余空间全部占用：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> grant select on dba_objects to scott;
Grant succeeded.
SYS@ora11g> conn scott/tiger
Connected.
SCOTT@ora11g> create table tab_obj as select * from dba_objects;
Table created.
SCOTT@ora11g> insert into tab_obj select * from tab_obj;
13524 rows created.
SCOTT@ora11g> /
27048 rows created.
SCOTT@ora11g> /
54096 rows created.
SCOTT@ora11g> /
108192 rows created.
SCOTT@ora11g> /
216384 rows created.
SCOTT@ora11g> /
432768 rows created.
SCOTT@ora11g> /
insert into tab_obj select * from tab_obj
*
ERROR at line 1:
ORA-01653: unable to extend table SCOTT.TAB_OBJ by 1024 in tablespace USERS
```

可见此时，users 表空间已经没有剩余空间了，那么 recyclebin 中的对象是否还存在呢？

```
SCOTT@ora11g> show recyclebin;
SCOTT@ora11g>
```

都不存在了。

可见，当数据文件没有打开自动扩展时，如果该数据文件中剩余空间不足，Oracle 会自动清除 recyclebin 中的对象来释放空间。那如果打开自动扩展呢？

```
SCOTT@ora11g> alter database datafile 4 autoextend on maxsize 10G;
alter database datafile 4 autoextend on maxsize 10G
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

权限不够，我们切换到 sys 用户：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> alter database datafile 4 autoextend on maxsize 10G;
Database altered.
```

我们重新做这个实验：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> select * from tab;
TNAME         TABTYPE CLUSTERID
------------- --------------------
BONUS         TABLE
DEPT          TABLE
EMP           TABLE
SALGRADE      TABLE
TAB_OBJ       TABLE

SCOTT@ora11g> drop table tab_obj;
Table dropped.
SCOTT@ora11g> show recyclebin;
ORIGINAL NAME RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
------------- ------------------------------ ------------ ----------------
TAB_OBJ       BIN$Mok/8djQGtfgU2/IqMCdMA==$0 TABLE        2016-05-11:11:00:53
SCOTT@ora11g> create table tab_obj as select * from dba_objects;
Table created.
SCOTT@ora11g> insert into tab_obj select * from tab_obj;
13524 rows created.
SCOTT@ora11g> /
27048 rows created.
SCOTT@ora11g> /
54096 rows created.
SCOTT@ora11g> /
108192 rows created.
SCOTT@ora11g> /
216384 rows created.
SCOTT@ora11g> /
432768 rows created.
SCOTT@ora11g> /
```

```
865536 rows created.
SCOTT@ora11g> show recyclebin;
SCOTT@ora11g>
```

此时，我们再去查看 users 表空间中 4 号文件的大小：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> select file_id,BYTES/1024/1024,AUTOEXTENSIBLE,MAXBYTES/1024/1024
from dba_data_files where file_id=4;
   FILE_ID BYTES/1024/1024 AUT MAXBYTES/1024/1024
---------- --------------- --- ------------------
         4        188.3125 YES              10240
```

因此，结论也就出来了：即便打开数据文件自动扩展，当数据文件中剩余空间不足时，Oracle 同样会清除 recyclebin 中的对象。

在网络上，经常会有人争论一些问题，此时，往往就会有人直接构造实验，通过实验来证明自己的论点，这显然是极有说服力的。在 Oracle 中也是如此，我们可以利用实验来完成两件事情：

第一，验证结论。我们可以根据已有的知识来得到某一观点或者结论。但它是否正确呢？我们可以依照假设，在满足前提条件的情况下构造一个实验，并完成该实验来得到结论，从而来证明我们的结论是否正确。

第二，探查未知。我们在学习过程中，经常会遇到困惑的地方，不知道再深入思考一下会得到怎样的结论。此时，也是我们可以动手的时候了。我们同样可以构造实验，依据实验结论来释疑解惑。

上面的闪回表删除，正是我们通过逐步构造实验的方法，来获得正确的结论。

接下来，我们看闪回查询。

Oracle 数据库中的闪回查询、闪回事务查询和闪回版本查询三种闪回技术，都是基于 undo 信息的。后两者都是基于闪回查询的变种而已，这里只详细介绍闪回查询的相关实验。其他两种读者可自行查阅相关资料。

我们先创建一个测试表：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create table tab_emp as select * from emp;
Table created.
```

然后查看当前的 scn 和时间：

```
SCOTT@ora11g> conn / as sysdba
Connected.
```

```
SYS@ora11g> select current_scn from v$database;
CURRENT_SCN
-----------
     397131
```

也可以使用系统包来获取当前的 scn：

```
SYS@ora11g> select dbms_flashback.get_system_change_number from dual;
GET_SYSTEM_CHANGE_NUMBER
------------------------
    397133
```

获取当前时间：

```
SYS@ora11g> select to_char(sysdate,'yyyy-mm-dd hh24:mi:ss') from dual;
TO_CHAR(SYSDATE,'YY
-------------------
2016-05-11 14:02:11
```

然后将 tab_emp 表中的数据删除。

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> delete from tab_emp;
14 rows deleted.
SCOTT@ora11g> commit;
Commit complete.
SCOTT@ora11g> select * from tab_emp;
no rows selected
```

此时，tab_emp 中已经没有任何数据了。但是，我们忽然想起来我们要删除的表不是 tab_emp……那问题来了，这个表中的数据已经被删除了，如何恢复？

我们回顾一下前面提到 undo 是修改数据时生成的“前镜像”，那么，也就是说，如果能够去找到 delete 操作时生成并存在于 undo 中的信息，显然就可以将这些数据再找回来。其实闪回查询也正是这样做的：

```
SCOTT@ora11g> set pages 100
SCOTT@ora11g> set line 100
SCOTT@ora11g> select * from tab_emp as of scn 397133;
    EMPNO ENAME      JOB          MGR HIREDATE      SAL    COMM     DEPTNO
--------- --------- ---------- ------------------ ----- -------- --------
     7369 SMITH      CLERK       7902 17-DEC-80     800     20
     7499 ALLEN      SALESMAN    7698 20-FEB-81    1600    300    30
```

```
    7521 WARD      SALESMAN   7698 22-FEB-81   1250   500   30
    7566 JONES     MANAGER    7839 02-APR-81   2975    20
    7654 MARTIN    SALESMAN   7698 28-SEP-81   1250  1400   30
    7698 BLAKE     MANAGER    7839 01-MAY-81   2850    30
    7782 CLARK     MANAGER    7839 09-JUN-81   2450    10
    7788 SCOTT     ANALYST    7566 19-APR-87   3000    20
    7839 KING      PRESIDENT       17-NOV-81   5000    10
    7844 TURNER    SALESMAN   7698 08-SEP-81   1500     0   30
    7876 ADAMS     CLERK      7788 23-MAY-87   1100    20
    7900 JAMES     CLERK      7698 03-DEC-81    950    30
    7902 FORD      ANALYST    7566 03-DEC-81   3000    20
    7934 MILLER    CLERK      7782 23-JAN-82   1300    10

14 rows selected.
```

上面使用 scn 来查看 delete 操作之前的数据，也可以使用时间戳：

```
SCOTT@ora11g> select * from tab_emp as of timestamp to_timestamp('2016-05-11 14:02:11','yyyy-mm-dd hh24:mi:ss');
```

输出结果省略。

显然，这样就可以找到 delete 操作之前，tab_emp 中原有的数据。再将这些数据插入到原表中就可以恢复数据了：

```
SCOTT@ora11g> insert into tab_emp select * from tab_emp as of scn 397133;
14 rows created.
SCOTT@ora11g> commit;
Commit complete.
```

也可以使用闪回表来处理：

```
SCOTT@ora11g> flashback table tab_emp to scn 397133;
flashback table tab_emp to scn 397133
              *
ERROR at line 1:
ORA-08189: cannot flashback the table because row movement is not enabled
```

这里报错，提示需要先启用行移动：

```
SCOTT@ora11g> alter table tab_emp enable row movement;
Table altered.
SCOTT@ora11g> flashback table tab_emp to scn 397133;
Flashback complete.
```

这样，我们就利用闪回查询将数据恢复回来了。当然，闪回查询是基于 undo 信息的，因此如果闪回查询依赖的 undo 信息已经被其他事务所生成的 undo 信息覆盖掉的话，则闪回查询就无法进行了。这也是闪回查询无法进行长时间闪回的原因。

再来看闪回数据库。

前面的几种闪回技术都对表级别的对象进行操作。闪回数据库则将整个数据库闪回到过去的某一个时间点或者 scn。比如，我们在生产系统中遇到了错误的数据，从而导致后续数据都出现问题，我们就需要将这段时间内系统的所有修改操作全部取消。或者我们对一个表执行了 truncate 操作，又想恢复这个表的数据。因为 truncate 基本不生成 undo 信息，因此无法回退。对于这些问题，我们都可以考虑使用闪回数据库技术进行恢复。

要使用闪回数据库，必须先开启闪回，而闪回又依赖于归档。因此又需要先开启归档。这些操作，我们已在本章前面已经全部完成。

一旦开启数据库闪回，Oracle 就会自动生成闪回日志：

```
[oracle@rhel6 ~]$ cd /u01/oracle/fra/ORA11G/flashback/
 [oracle@rhel6 flashback]$ ls
o1_mf_cm32jph9_.flb  o1_mf_cm32k32r_.flb
```

Oracle 提供了 v$flashback_database_log 视图，可用来查看能将数据库闪回到的最老的 scn 或者时间点：

```
SYS@ora11g> select OLDEST_FLASHBACK_SCN,
to_char(OLDEST_FLASHBACK_TIME,'yyyy-mm-dd hh24:mi:ss')
from v$flashback_database_log;
OLDEST_FLASHBACK_SCN TO_CHAR(OLDEST_FLAS
-------------------- -------------------
             306942  2016-05-10 15:14:30
```

我们尝试将一个表 truncate，然后利用闪回数据库技术进行恢复：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create table tab_test as select * from user_objects;
Table created.
SCOTT@ora11g> select count(*) from tab_test;
  COUNT(*)
----------
 10
```

然后查看当前时间：

```
SCOTT@ora11g> select to_char(sysdate,'yyyy-mm-dd hh24:mi:ss') from dual;
```

```
TO_CHAR(SYSDATE,'YY
-------------------
2016-05-11 14:48:23
```

接下来 truncate 该表：

```
SCOTT@ora11g> truncate table tab_test;
Table truncated.
SCOTT@ora11g> select count(*) from tab_test;
  COUNT(*)
----------
  0
```

然后将数据库重启到 mount 阶段并闪回：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SYS@ora11g> startup mount;
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size      2257840 bytes
Variable Size    553651280 bytes
Database Buffers  276824064 bytes
Redo Buffers      2371584 bytes
Database mounted.
SYS@ora11g> flashback database to timestamp to_timestamp('2016-05-11
14:48:23','yyyy-mm-dd hh24:mi:ss');
Flashback complete.
```

闪回结束后，我们以 read only 方式打开数据库，这样我们可以预防其他用户连接到数据库来修改数据，我们自己可以验证被 truncate 的表，看数据是否已经恢复回来：

```
SYS@ora11g> alter database open read only;
Database altered.
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> select count(*) from tab_test;
  COUNT(*)
```

```
----------
 10
```

然后，再以正常方式重启数据库：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> startup force;
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size      2257840 bytes
Variable Size    553651280 bytes
Database Buffers  276824064 bytes
Redo Buffers     2371584 bytes
Database mounted.
ORA-01589: must use RESETLOGS or NORESETLOGS option for database open
```

这里报错，我们用 resetlogs 方式打开数据库：

```
SYS@ora11g> alter database open resetlogs;
Database altered.
```

需要注意，闪回数据库操作可以重复进行。也就是说，你将数据库启动到 mount 后，可以进行多次闪回，从而将数据库恢复到你需要的时间点。

最后一个是闪回归档。

该特性从 11g 版本开始支持。在实际系统中，我们可能遇到这样的需求：有几个存放敏感数据的表，我们期望将这些表上的每次增删改操作都记录下来，这样在以后可以进行审计或者复查。从某种意义上说，这样的需求，数据库中的审计和触发器功能都可以实现。但闪回归档更简单一些。审计和触发器相关知识，我们将在后续书籍中进行说明。

来看实验。

先创建用来存储闪回归档的表空间。

```
SYS@ora11g> create tablespace tbs_fra datafile
'/u01/oracle/oradata/ora11g/tbs_fra.dbf' size 50M autoextend on maxsize 10G;
Tablespace created.
```

然后创建闪回归档管理用户并授权。

```
SYS@ora11g> create user u_fra identified by fra default tablespace tbs_fra;
User created.
SYS@ora11g> grant connect,resource,flashback archive administer to u_fra;
Grant succeeded.
```

接下来创建闪回归档。

```
SYS@ora11g> conn u_fra/fra

Connected.
U_FRA@ora11g> create flashback archive fra1 tablespace tbs_fra retention 10
year;
Flashback archive created.
```

然后针对需要归档的表，开启闪回归档：

```
U_FRA@ora11g> alter table scott.tab_test flashback archive fra1;
alter table scott.emp flashback archive fra1
*
ERROR at line 1:
ORA-00942: table or view does not exist
```

这里缺少权限，我们授权，然后再来。

```
U_FRA@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g>grant all on tab_test to u_fra;
Grant succeeded.
SCOTT@ora11g> conn u_fra/fra
Connected.
U_FRA@ora11g> alter table scott.tab_test flashback archive fra1;
Table altered.
```

这样，就完成了对表 tab_test 开启闪回归档的操作。

此时，我们将 tab_test 表中的数据全部删除：

```
SCOTT@ora11g> delete tab_test;
10 rows deleted.
SCOTT@ora11g> commit;
Commit complete.
```

然后查看 scott 用户的表：

```
SCOTT@ora11g> select * from tab;
TNAME                          TABTYPE CLUSTERID
------------------------------ ----------
BONUS                          TABLE
DEPT                           TABLE
EMP                            TABLE
```

```
SALGRADE                    TABLE
SYS_FBA_DDL_COLMAP_13841    TABLE
SYS_FBA_HIST_13841          TABLE
SYS_FBA_TCRV_13841          TABLE
SYS_TEMP_FBT                TABLE
TAB_EMP                     TABLE
TAB_OBJ                     TABLE
TAB_TEST                    TABLE

11 rows selected.
```

这里出现了一些 sys_fba 打头的对象，这些表记录了 tab_test 中数据变化的历史信息。这样，我们就可以对表 tab_test 进行闪回查询，并且该查询不受 undo 信息保存时间的限制，而只取决于闪回归档的保留设置。

关于闪回技术的详细描述，也可以参考官方文档 Database Advanced Application Developer's Guide 中的第 12 章 Using Oracle Flashback Technology。

5.11 基于表空间的时间点恢复实验

基于表空间的时间点恢复，也叫 TSPITR(Tablespace Point-In-Time Recovery)。就是将一个或者多个表空间打包恢复到过去的某一个 scn 或者时间点，而数据库中其他表空间中的对象保持不变。该技术适用于如下场景：

1) 逻辑上独立的表空间，该表空间中重要的表中数据被删除或者修改有误。
2) 错误使用 DDL 操作更改了某个表空间中的表的结构，这无法使用闪回来修复。
3) 删除表时使用了 purge 选项。

当然，它也有限制，被重命名过的表空间，或者已经被删除的表空间则无法使用此项技术。

我们来看一个完整实验。

先创建测试表空间。

```
SYS@ora11g> create tablespace tbs_pitr datafile
'/u01/oracle/oradata/ora11g/tbs_pitr.dbf' size 50M;
Tablespace created.
SYS@ora11g> create tablespace tbs_pitr_I datafile
'/u01/oracle/oradata/ora11g/tbs_pitr_i.dbf' size 50M;
Tablespace created.
```

然后创建测试用户及测试表。

```
SYS@ora11g> create user u_pr identified by pr default tablespace tbs_pitr;
```

```
User created.
SYS@ora11g> grant connect,resource to u_pr;
Grant succeeded.
SYS@ora11g> conn u_pr/pr;
Connected.
U_PR@ora11g> create table test (id number,name varchar2(30));
Table created.
U_PR@ora11g> create index ind_id on test(id) tablespace tbs_pitr_i;
Index created.
U_PR@ora11g> insert into test values (1,1);
1 row created.
U_PR@ora11g> insert into test values (2,2);
1 row created.
U_PR@ora11g> commit;
Commit complete.
```

这样，我们创建的表和索引分别在两个表空间上。也就是说，这两个表空间中的对象存在依赖关系。

接下来对数据库进行备份：

```
[oracle@rhel6 ~]$ rman target / catalog u_rcadm/admin@catdb
Recovery Manager: Release 11.2.0.4.0 - Production on Wed May 11 15:46:31 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
connected to target database: ORA11G (DBID=12479381)
connected to recovery catalog database
RMAN> backup database plus archivelog;
```

输出结果省略。

然后记录下当前时间，并对表 test 进行 truncate 操作：

```
U_PR@ora11g> select to_char(sysdate,'yyyy-mm-dd hh24:mi:ss') from dual;
TO_CHAR(SYSDATE,'YY
-------------------
2016-05-11 15:48:14
U_PR@ora11g> truncate table test;
Table truncated.
```

接下来，我们使用 TSPITR 将数据恢复回来。

先查看这两个表空间中是否存在彼此依赖的对象：

```
SYS@ora11g> col obj1_name for a10
SYS@ora11g> col ts1_name for a10
```

```
SYS@ora11g> col obj2_name for a10
SYS@ora11g> col ts2_name for a10
SYS@ora11g> select OBJ1_NAME,TS1_NAME,OBJ2_NAME,TS2_NAME
from sys.ts_pitr_check
where (ts1_name in ('TBS_PITR') and ts2_name not in ('TBS_PITR'))
or (ts1_name not in ('TBS_PITR_I') and ts2_name in ('TBS_PITR_I'));
OBJ1_NAME  TS1_NAME   OBJ2_NAME  TS2_NAME
----------- ---------  ---------- ----------
TEST        TBS_PITR   IND_ID  TBS_PITR_I
```

根据上面的结果，可以知道两个表空间中，表 test 和索引 ind_id 分别位于不同的表空间中，并且它们之间存在依赖关系。也就是说，这两个表空间需要同时进行恢复。那么问题来了，如果将这两个表空间都进行恢复，而这两个表空间中还有其他对象，这些对象是不是就可能会丢失？尤其是那些刚好在我们要恢复的时间点之后创建的对象。我们先查询这些对象：

```
SYS@ora11g> select owner, name, tablespace_name,
       to_char(creation_time, 'yyyy-mm-dd:hh24:mi:ss')
  from sys.ts_pitr_objects_to_be_dropped
 where tablespace_name in ('TBS_PITR', 'TBS_PITR_I')
   and creation_time >
       to_date('2016-05-11 15:48:14', 'yyyy-mm-dd:hh24:mi:ss')
 order by tablespace_name, creation_time;
no rows selected
```

因为我们再没有在这两个表空间中创建其他对象，因此查询结果为空。在实际环境中，可能会查询出来一些对象，那时你需要考虑如何处理这些对象。是先使用 exp 导出，待表空间恢复完成之后再导入，还是可以直接舍弃，取决于你的实际环境了。

接下来，将这两个表空间置于离线状态：

```
SYS@ora11g> alter tablespace tbs_pitr offline;
Tablespace altered.
SYS@ora11g> alter tablespace tbs_pitr_i offline;
Tablespace altered.
```

然后我们开始恢复。先创建一个目录，这在后面的恢复中需要用到。

```
[oracle@rhel6 ~]$ mkdir -p /home/oracle/pitr
[oracle@rhel6 ~]$ export NLS_LANG=AMERICAN_AMERICA.AL32UTF8
[oracle@rhel6 ~]$ export NLS_DATE_FORMAT='YYYY-MM-DD HH24:MI:SS'
[oracle@rhel6 ~]$ rman target / catalog u_rcadm/admin@catdb
Recovery Manager: Release 11.2.0.4.0 - Production on Wed May 11 16:06:11 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
```

```
connected to target database: ORA11G (DBID=12479381)
connected to recovery catalog database
RMAN> sql 'alter session set nls_date_format="yyyy-mm-dd hh24:mi:ss"';
starting full resync of recovery catalog
full resync complete
sql statement: alter session set nls_date_format="yyyy-mm-dd hh24:mi:ss"
```

执行 TSPITR：

```
RMAN> recover tablespace tbs_pitr,tbs_pitr_i until time '2016-05-11 15:48:14'
auxiliary destination '/home/oracle/pitr';
```

这里输出内容极长，我们省略。

在恢复过程中，我们查看 pitr 目录：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ ls
backup  create_db.sql  dump1.dmp  pitr  temp002.dbf
[oracle@rhel6 ~]$ cd pitr/
[oracle@rhel6 pitr]$ ls
ORA11G  tspitr_ElbB_11608.dmp
[oracle@rhel6 pitr]$ cd ORA11G/
[oracle@rhel6 ORA11G]$ ls
controlfile  datafile  onlinelog
```

实际上，基于表空间的时间点恢复，实质上就是将整个数据库先恢复到指定时间点，恢复出来的内容都先暂且存储在指定位置(这里就是 pitr 目录)，然后将表空间中的数据从临时库中导出，最后导入起初的库中，从而完成恢复。

恢复完成后，先将这两个表空间重置为 online 状态：

```
SYS@ora11g> alter tablespace tbs_pitr online;
Tablespace altered.
SYS@ora11g> alter tablespace tbs_pitr_i online;
Tablespace altered.
```

然后再检查数据：

```
SYS@ora11g> conn u_pr/pr;
Connected.
U_PR@ora11g> select * from test;
 ID NAME
---------- ------------------------------
        1 1
        2 2
```

可见，数据都已经恢复回来了。

5.12 数据库手工备份实验

本章做的实验，主要是利用数据库自身的特性或者功能进行备份或者恢复。但设想一下，如果数据库没有开启归档，或者说我们期望能够使用 cp 这样的命令对数据库进行一次完全备份，则应如何处理？

数据库没有开启归档的话，则无法使用 RMAN 进行备份。这一点读者可以自行验证。另外，没有开启归档，则也无法在 open 状态下对数据库进行备份。因此，这里的备份只能在数据库关闭的状态下进行。当我们要对数据库进行升级或者执行类似的大动作时，手工冷备是一种快捷且保险的备份方式。所谓冷备，就是在数据库关闭的状态下进行的备份。

接下来，我们就对数据库做一次完全冷备。

第一件事情，就是先要确定备份哪些文件。Oracle 数据库中最关键的文件是数据文件、日志文件、控制文件以及参数文件。我们这里假设数据库没有开启归档，因此归档日志不在考虑范围之内。也就是说，只要我们备份了这四类文件，就可以完成数据库恢复。

确定了要备份这四类文件，我们就需要知道这些文件的名称及存放何处了。我们可以执行如下查询：

```
U_PR@ora11g> conn / as sysdba
Connected.
SYS@ora11g> select name from v$datafile;
NAME
---------------------------------------------------------------------------
/u01/oracle/oradata/ora11g/system01.dbf
/u01/oracle/oradata/ora11g/sysaux01.dbf
/u01/oracle/oradata/ora11g/undotbs01.dbf
/u01/oracle/oradata/ora11g/users01.dbf
/u01/oracle/oradata/ora11g/tbs_test_01.dbf
/u01/oracle/oradata/ora11g/tbs_test_02.dbf
/u01/oracle/oradata/ora11g/tbs_fra.dbf
/u01/oracle/oradata/ora11g/tbs_pitr.dbf
/u01/oracle/oradata/ora11g/tbs_pitr_i.dbf
9 rows selected.

SYS@ora11g> select member from v$logfile;
MEMBER
---------------------------------------------------------------------------
/u01/oracle/oradata/ora11g/redo04a.log
```

```
/u01/oracle/fra/redo04b.log
/u01/oracle/oradata/ora11g/redo05a.log
/u01/oracle/fra/redo05b.log
/u01/oracle/oradata/ora11g/redo06a.log
/u01/oracle/fra/redo06b.log
6 rows selected.

SYS@ora11g> select name from v$controlfile;
NAME
--------------------------------------------------------------------------
/u01/oracle/oradata/ora11g/control01.ctl
/u01/oracle/fra/control2.ctl
/home/oracle/backup/control/control03.ctl

SYS@ora11g> select value from v$parameter where name='spfile';
VALUE
--------------------------------------------------------------------------
/u01/oracle/product/11.2.0/dbs/spfileora11g.ora
```

这样，我们就查询出了要备份的所有文件。接下来创建一个文件，将这些文件名称都存放进去，并添加 tar 命令，如下：

```
[oracle@rhel6 ~]$ vi backup_db.sh
tar -zcvf db_cold_bak.tar
/u01/oracle/oradata/ora11g/system01.dbf
/u01/oracle/oradata/ora11g/sysaux01.dbf
/u01/oracle/oradata/ora11g/undotbs01.dbf
/u01/oracle/oradata/ora11g/users01.dbf
/u01/oracle/oradata/ora11g/tbs_test_01.dbf
/u01/oracle/oradata/ora11g/tbs_test_02.dbf
/u01/oracle/oradata/ora11g/tbs_fra.dbf
/u01/oracle/oradata/ora11g/tbs_pitr.dbf
/u01/oracle/oradata/ora11g/tbs_pitr_i.dbf
/u01/oracle/oradata/ora11g/redo04a.log
/u01/oracle/fra/redo04b.log
/u01/oracle/oradata/ora11g/redo05a.log
/u01/oracle/fra/redo05b.log
/u01/oracle/oradata/ora11g/redo06a.log
/u01/oracle/fra/redo06b.log
/u01/oracle/oradata/ora11g/control01.ctl
/u01/oracle/fra/control2.ctl
```

```
/home/oracle/backup/control/control03.ctl
/u01/oracle/product/11.2.0/dbs/spfileora11g.ora
```

但这个文件中有太多换行符，我们需要把这些换行符都去掉才能执行这个脚本。可将光标放在该文件中第一行的末尾处，记住要跟前面的内容空出来几个空格，然后按住大写 J，则会自动删除所有换行符。执行完毕后内容可看到如下：

```
tar -zcvf db_cold_bak.tar   /u01/oracle/oradata/ora11g/system01.dbf
/u01/oracle/oradata/ora11g/sysaux01.dbf
/u01/oracle/oradata/ora11g/undotbs01.dbf /u01/oracle/oradata/ora11g/users01.dbf
/u01/oracle/oradata/ora11g/tbs_test_01.dbf
/u01/oracle/oradata/ora11g/tbs_test_02.dbf
/u01/oracle/oradata/ora11g/tbs_fra.dbf /u01/oracle/oradata/ora11g/tbs_pitr.dbf
/u01/oracle/oradata/ora11g/tbs_pitr_i.dbf
/u01/oracle/oradata/ora11g/redo04a.log /u01/oracle/fra/redo04b.log
/u01/oracle/oradata/ora11g/redo05a.log /u01/oracle/fra/redo05b.log
/u01/oracle/oradata/ora11g/redo06a.log /u01/oracle/fra/redo06b.log
/u01/oracle/oradata/ora11g/control01.ctl /u01/oracle/fra/control2.ctl
/home/oracle/backup/control/control03.ctl
/u01/oracle/product/11.2.0/dbs/spfileora11g.ora
```

然后保存退出。

接下来关闭数据库：

```
SYS@ora11g> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
```

然后执行备份：

```
[oracle@rhel6 ~]$ sh backup_db.sh
```

输出结果省略。

这里生成的 tar 文件就是数据库中所有文件的备份。

```
[oracle@rhel6 ~]$ ls
backup  backup_db.sh  create_db.sql  db_cold_bak.tar  dump1.dmp  pitr
temp002.dbf
```

当我们想恢复数据库时，直接解压该 tar 包并将所有文件放到原位置即可。

5.13　数据库灾难恢复实验

所谓灾难恢复指的是数据库的数据文件、日志文件、控制文件以及参数文件都遭到不同程度的破坏，我们需要使用备份进行全库恢复。可使用 5.12 一节创建的冷备进行恢复。也可以使用 RMAN 生成的备份进行恢复。

使用冷备进行全库恢复比较简单，只需要执行 tar –zxvf db_cold_bak.tar –C/。

这样的命令就可以将 5.12 一节中生成的备份文件直接全部恢复到其原来的位置，并启动数据库即可。因此这里只描述如何使用 RMAN 备份对数据库进行完全恢复。

先启动 target 数据库 ora11g，这个是我们将要模拟故障并进行恢复的数据库：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Thu May 12 09:06:37 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SYS@ora11g> startup
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size          2257840 bytes
Variable Size     553651280 bytes
Database Buffers 276824064 bytes
Redo Buffers        2371584 bytes
Database mounted.
Database opened.
```

然后启动 catalog 数据库 catdb：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ export ORACLE_SID=catdb
[oracle@rhel6 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.4.0 Production on Thu May 12 09:07:00 2016
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to an idle instance.
SYS@catdb> startup
ORACLE instance started.
Total System Global Area  835104768 bytes
Fixed Size          2257840 bytes
Variable Size     541068368 bytes
Database Buffers 289406976 bytes
Redo Buffers        2371584 bytes
```

```
Database mounted.
Database opened.
```

接下来启动监听并查看其状态，这样我们就可以使用 RMAN 通过监听同时连接到 target 和 catalog 数据库：

```
[root@rhel6 Desktop]# su - oracle
[oracle@rhel6 ~]$ lsnrctl start
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 12-MAY-2016 09:07:19
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Starting /u01/oracle/product/11.2.0/bin/tnslsnr: please wait...
TNSLSNR for Linux: Version 11.2.0.4.0 - Production
System parameter file is
/u01/oracle/product/11.2.0/network/admin/listener.ora
Log messages written to /u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date                12-MAY-2016 09:07:21
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/oracle/product/11.2.0/network/admin/listener.ora
Listener Log File
/u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
The listener supports no services
The command completed successfully

[oracle@rhel6 ~]$ lsnrctl status
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 12-MAY-2016 09:09:13
Copyright (c) 1991, 2013, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel6)(PORT=1521)))
```

```
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date                12-MAY-2016 09:07:21
Uptime                    0 days 0 hr. 1 min. 52 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/oracle/product/11.2.0/network/admin/listener.ora
Listener Log File
/u01/oracle/diag/tnslsnr/rhel6/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel6)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
Services Summary...
Service "catdb" has 1 instance(s).
  Instance "catdb", status READY, has 1 handler(s) for this service...
Service "catdbXDB" has 1 instance(s).
  Instance "catdb", status READY, has 1 handler(s) for this service...
Service "ora11g" has 1 instance(s).
  Instance "ora11g", status READY, has 4 handler(s) for this service...
The command completed successfully
```

接下来登录到 RMAN，将原备份全部删除，重新生成一个完整备份：

```
[oracle@rhel6 ~]$ rman target sys/oracle@ora11g catalog u_rcadm/admin@catdb
Recovery Manager: Release 11.2.0.4.0 - Production on Thu May 12 09:17:28 2016
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
connected to target database: ORA11G (DBID=12479381)
connected to recovery catalog database
RMAN> delete noprompt backup;
```

输出结果略。

```
RMAN> backup database plus archivelog;
```

同样省略输出结果。

接下来对目标数据库 ora11g 做一些破坏，从而模拟故障：

```
[oracle@rhel6 ~]$ cd /u01/oracle/oradata/ora11g/
```

```
[oracle@rhel6 ora11g]$ ls
control01.ctl  redo06a.log  tbs_fra.dbf    tbs_test_01.dbf  temp_new.dbf
redo04a.log   sysaux01.dbf  tbs_pitr.dbf   tbs_test_02.dbf  undotbs01.dbf
redo05a.log   system01.dbf  tbs_pitr_i.dbf  temp001.dbf     users01.dbf
[oracle@rhel6 ora11g]$ rm *
[oracle@rhel6 ora11g]$ cd /u01/oracle/fra/
[oracle@rhel6 fra]$ ls
control2.ctl  ORA11G  redo04b.log  redo05b.log  redo06b.log
[oracle@rhel6 fra]$ rm control2.ctl
[oracle@rhel6 fra]$ rm redo0*
[oracle@rhel6 fra]$ cd $ORACLE_HOME/dbs
[oracle@rhel6 dbs]$ rm *ora11g.ora
[oracle@rhel6 dbs]$ cd
[oracle@rhel6 ~]$ rm temp002.dbf
[oracle@rhel6 ~]$ cd backup
[oracle@rhel6 backup]$ ls
control  tbs01.bak  tbs02.bak
[oracle@rhel6 backup]$ cd control/
[oracle@rhel6 control]$ rm control03.ctl
```

这样，就将 ora11g 数据库所有的数据文件、联机日志文件、控制文件、参数文件(包含 pfile 和 spfile)以及临时文件全部删除了。

接下来，我们触发故障：

```
SYS@ora11g> alter system checkpoint;
System altered.
SYS@ora11g> shutdown immediate;
Database closed.
ORA-00210: cannot open the specified control file
ORA-00202: control file: '/u01/oracle/oradata/ora11g/control01.ctl'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

显然，此时 Oracle 已经检测到数据库出现了问题。我们以 abort 方式关闭数据库并重新启动：

```
SYS@ora11g> shutdown abort;
ORACLE instance shut down.
SYS@ora11g> startup
ORA-01078: failure in processing system parameters
```

```
    LRM-00109: could not open parameter file
'/u01/oracle/product/11.2.0/dbs/initora11g.ora'
```

检测到参数文件丢失。

我们进入 RMAN 来进行恢复：

```
    RMAN> exit
    RMAN-06900: WARNING: unable to generate V$RMAN_STATUS or V$RMAN_OUTPUT row
    RMAN-06901: WARNING: disabling update of the V$RMAN_STATUS and V$RMAN_OUTPUT
rows
    ORACLE error from target database:
    ORA-03113: end-of-file on communication channel
    Process ID: 4026
    Session ID: 38 Serial number: 13
    Recovery Manager complete.

    [oracle@rhel6 ~]$ rman target / catalog u_rcadm/admin@catdb
    Recovery Manager: Release 11.2.0.4.0 - Production on Thu May 12 09:32:59 2016
    Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
    connected to target database (not started)
    connected to recovery catalog database
    RMAN>
```

按照前面 4.7 一节中数据库启动三个阶段中的描述，我们需要先恢复参数文件和控制文件。先强制启动一个空实例：

```
    RMAN> startup nomount force;
    startup failed: ORA-01078: failure in processing system parameters
    LRM-00109: could not open parameter file
'/u01/oracle/product/11.2.0/dbs/initora11g.ora'
    starting Oracle instance without parameter file for retrieval of spfile
    Oracle instance started
    Total System Global Area  1068937216 bytes
    Fixed Size                2260088 bytes
    Variable Size             281019272 bytes
    Database Buffers          780140544 bytes
    Redo Buffers              5517312 bytes
```

然后从自动备份中还原 spfile：

```
    RMAN> restore spfile from autobackup;
    Starting restore at 12-MAY-16
```

```
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=19 device type=DISK
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160512
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160511
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160510
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160509
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160508
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160507
channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160506
channel ORA_DISK_1: no AUTOBACKUP in 7 days found
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of restore command at 05/12/2016 09:36:41
RMAN-06172: no AUTOBACKUP found or specified handle is not a valid copy or piece
```

这里报错，Oracle 指出没有找到自动备份，我们来查看一下：

```
RMAN> show CONTROLFILE AUTOBACKUP;
RMAN configuration parameters for database with db_unique_name DUMMY are:
CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

显然，在 RMAN 中我们确实配置了控制文件自动备份。但注意上面内容中的阴影着重显示部分，前面启动的是一个 Oracle 自己提供的一个实例，这个实例所属的数据库称为 DUMMY，而不是 ora11g。因此，这里需要找到包含 ora11g 的 spfile 的备份文件：

```
[oracle@rhel6 ORA11G]$ cd /u01/oracle/fra/ORA11G/
[oracle@rhel6 ORA11G]$ cd autobackup/
[oracle@rhel6 autobackup]$ ls
2016_05_10  2016_05_11  2016_05_12
[oracle@rhel6 autobackup]$ cd 2016_05_12/
[oracle@rhel6 2016_05_12]$ ls
o1_mf_s_911639923_cm7phmm5_.bkp
```

这个文件就是前面备份时生成的包含 spfile 和控制文件的备份文件，我们使用该文件来还原 spfile：

```
RMAN>
restore spfile from
'/u01/oracle/fra/ORA11G/autobackup/2016_05_12/o1_mf_s_911639923_cm7phmm5_.bkp';
Starting restore at 12-MAY-16
using channel ORA_DISK_1
```

```
    channel ORA_DISK_1: restoring spfile from AUTOBACKUP
/u01/oracle/fra/ORA11G/autobackup/2016_05_12/o1_mf_s_911639923_cm7phmm5_.bkp
    channel ORA_DISK_1: SPFILE restore from AUTOBACKUP complete
    Finished restore at 12-MAY-16
```

这样，就完成了 spfile 的还原，然后我们重新启动实例到 nomount 状态：

```
RMAN> startup force nomount;
Oracle instance started
Total System Global Area    835104768 bytes
Fixed Size                  2257840 bytes
Variable Size               553651280 bytes
Database Buffers            276824064 bytes
Redo Buffers                2371584 bytes
```

因为此时启动的是 ora11g 的实例，所以我们可以使用控制文件自动备份来还原控制文件：

```
    RMAN> restore controlfile from autobackup;
    Starting restore at 12-MAY-16
    allocated channel: ORA_DISK_1
    channel ORA_DISK_1: SID=19 device type=DISK
    recovery area destination: /u01/oracle/fra
    database name (or database unique name) used for search: ORA11G
    channel ORA_DISK_1: AUTOBACKUP
/u01/oracle/fra/ORA11G/autobackup/2016_05_12/o1_mf_s_911639923_cm7phmm5_.bkp
found in the recovery area
    channel ORA_DISK_1: looking for AUTOBACKUP on day: 20160512
    channel ORA_DISK_1: restoring control file from AUTOBACKUP
/u01/oracle/fra/ORA11G/autobackup/2016_05_12/o1_mf_s_911639923_cm7phmm5_.bkp
    channel ORA_DISK_1: control file restore from AUTOBACKUP complete
    output file name=/u01/oracle/oradata/ora11g/control01.ctl
    output file name=/u01/oracle/fra/control2.ctl
    output file name=/home/oracle/backup/control/control03.ctl
    Finished restore at 12-MAY-16
```

然后，将数据库启动到 mount 状态：

```
RMAN> alter database mount;
database mounted
released channel: ORA_DISK_1
```

此时，我们可利用前面 5.8 一节中的自动修复选项，来尝试进行数据库恢复：

```
RMAN> list failure;
List of Database Failures
=========================
Failure ID Priority Status  Time Detected Summary
---------- -------- ------- ------------- -------
1302       HIGH     OPEN     11-MAY-16  Tablespace 11: 'TBS_PITR' is offline
1274       HIGH     OPEN     11-MAY-16  Tablespace 12: 'TBS_PITR_I' is offline
1256       HIGH     OPEN     11-MAY-16  One or more non-system datafiles need
media recovery
```

好吧，看似前面 5.11 一节中的实验尚未结束，这里居然还检测到 tbs_pitr 和 tbs_pitr_i 两个表空间状态是 offline。没关系，我们一并进行处理。

```
RMAN> advise failure;
Starting implicit crosscheck backup at 12-MAY-16
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=19 device type=DISK
Crosschecked 3 objects
Finished implicit crosscheck backup at 12-MAY-16
Starting implicit crosscheck copy at 12-MAY-16
using channel ORA_DISK_1
Crosschecked 4 objects
Finished implicit crosscheck copy at 12-MAY-16
searching for all files in the recovery area
cataloging files...
cataloging done

List of Cataloged Files
=======================
File Name:
/u01/oracle/fra/ORA11G/autobackup/2016_05_12/o1_mf_s_911639923_cm7phmm5_.bkp
WARNING: new failures were found since last LIST FAILURE command
List of Database Failures
=========================
Failure ID Priority Status  Time Detected Summary
---------- -------- ------- ------------- -------
1473     CRITICAL  OPEN     12-MAY-16   System datafile 1:
'/u01/oracle/oradata/ora11g/system01.dbf' is missing
1470     CRITICAL  OPEN     12-MAY-16   Control file needs media recovery
642      HIGH      OPEN     12-MAY-16   One or more non-system datafiles are
missing
```

```
    1302     HIGH       OPEN     11-MAY-16   Tablespace 11: 'TBS_PITR' is offline
    1274     HIGH       OPEN     11-MAY-16   Tablespace 12: 'TBS_PITR_I' is offline
    1256     HIGH       OPEN     11-MAY-16   One or more non-system datafiles need
media recovery

    analyzing automatic repair options; this may take some time
    using channel ORA_DISK_1
    analyzing automatic repair options complete

    Not all specified failures can currently be repaired.
    The following failures must be repaired before advise for others can be given.

    Failure ID Priority Status Time Detected Summary
    ---------- --------- ------ --- -------------------
    1473       CRITICAL OPEN    12-MAY-16    System datafile 1:
'/u01/oracle/oradata/ora11g/system01.dbf' is missing
    1470       CRITICAL OPEN    12-MAY-16    Control file needs media recovery
    642        HIGH     OPEN    12-MAY-16    One or more non-system datafiles are
missing
    1256       HIGH     OPEN    11-MAY-16    One or more non-system datafiles need
media recovery

    Mandatory Manual Actions
    ========================
    no manual actions available

    Optional Manual Actions
    =======================
    1. If you have the correct version of the control file, then shutdown the database
and replace the old control file
    2. If file /u01/oracle/oradata/ora11g/system01.dbf was unintentionally renamed
or moved, restore it
    3. If file /u01/oracle/oradata/ora11g/sysaux01.dbf was unintentionally renamed
or moved, restore it
    4. If file /u01/oracle/oradata/ora11g/undotbs01.dbf was unintentionally
renamed or moved, restore it
    5. If file /u01/oracle/oradata/ora11g/users01.dbf was unintentionally renamed
or moved, restore it
    6. If file /u01/oracle/oradata/ora11g/tbs_test_01.dbf was unintentionally
renamed or moved, restore it
```

```
    7. If file /u01/oracle/oradata/ora11g/tbs_test_02.dbf was unintentionally
renamed or moved, restore it
    8. If file /u01/oracle/oradata/ora11g/tbs_fra.dbf was unintentionally renamed
or moved, restore it
    9. If file /u01/oracle/oradata/ora11g/tbs_pitr.dbf was unintentionally renamed
or moved, restore it
    10. If file /u01/oracle/oradata/ora11g/tbs_pitr_i.dbf was unintentionally
renamed or moved, restore it
    11. If you restored the wrong version of data file
/u01/oracle/oradata/ora11g/tbs_pitr.dbf, then replace it with the correct one
    12. If you restored the wrong version of data file
/u01/oracle/oradata/ora11g/tbs_pitr_i.dbf, then replace it with the correct one

    Automated Repair Options
    ========================
    Option Repair Description
    ------ ------------------
    1      Perform incomplete database recovery
      Strategy: The repair includes point-in-time recovery with some data loss
      Repair script: /u01/oracle/diag/rdbms/ora11g/ora11g/hm/reco_1737002323.hm
```

很好，这里提供了自动修复的脚本，我们看一下内容：

```
    [oracle@rhel6 ~]$ more
/u01/oracle/diag/rdbms/ora11g/ora11g/hm/reco_1737002323.hm
       # database point-in-time recovery until a missing log
       restore database until scn 430621;
       recover database until scn 430621;
       alter database open resetlogs;
```

我们利用该脚本进行恢复：

```
RMAN> repair failure;
Strategy: The repair includes point-in-time recovery with some data loss
Repair script: /u01/oracle/diag/rdbms/ora11g/ora11g/hm/reco_1737002323.hm
contents of repair script:
   # database point-in-time recovery until a missing log
   restore database until scn 430621;
   recover database until scn 430621;
   alter database open resetlogs;
Do you really want to execute the above repair (enter YES or NO)?
```

我们输入 yes，然后回车继续：

```
    executing repair script
    Starting restore at 12-MAY-16
    using channel ORA_DISK_1
    channel ORA_DISK_1: starting datafile backup set restore
    channel ORA_DISK_1: specifying datafile(s) to restore from backup set
    channel ORA_DISK_1: restoring datafile 00001 to
/u01/oracle/oradata/ora11g/system01.dbf
    channel ORA_DISK_1: restoring datafile 00002 to
/u01/oracle/oradata/ora11g/sysaux01.dbf
    channel ORA_DISK_1: restoring datafile 00003 to
/u01/oracle/oradata/ora11g/undotbs01.dbf
    channel ORA_DISK_1: restoring datafile 00004 to
/u01/oracle/oradata/ora11g/users01.dbf
    channel ORA_DISK_1: restoring datafile 00005 to
/u01/oracle/oradata/ora11g/tbs_test_01.dbf
    channel ORA_DISK_1: restoring datafile 00006 to
/u01/oracle/oradata/ora11g/tbs_test_02.dbf
    channel ORA_DISK_1: restoring datafile 00007 to
/u01/oracle/oradata/ora11g/tbs_fra.dbf
    channel ORA_DISK_1: restoring datafile 00008 to
/u01/oracle/oradata/ora11g/tbs_pitr.dbf
    channel ORA_DISK_1: restoring datafile 00009 to
/u01/oracle/oradata/ora11g/tbs_pitr_i.dbf
    channel ORA_DISK_1: reading from backup piece
/u01/oracle/fra/ORA11G/backupset/2016_05_12/o1_mf_nnndf_TAG20160512T091813_cm7p
gqy3_.bkp
    channel ORA_DISK_1: piece
handle=/u01/oracle/fra/ORA11G/backupset/2016_05_12/o1_mf_nnndf_TAG20160512T0918
13_cm7pgqy3_.bkp tag=TAG20160512T091813
    channel ORA_DISK_1: restored backup piece 1
    channel ORA_DISK_1: restore complete, elapsed time: 00:00:25
    Finished restore at 12-MAY-16
    Starting recover at 12-MAY-16
    using channel ORA_DISK_1
    starting media recovery
    archived log for thread 1 with sequence 6 is already on disk as file
/u01/oracle/fra/ORA11G/archivelog/2016_05_12/o1_mf_1_6_cm7phjlk_.arc
    archived log file
name=/u01/oracle/fra/ORA11G/archivelog/2016_05_12/o1_mf_1_6_cm7phjlk_.arc
```

```
thread=1 sequence=6
    media recovery complete, elapsed time: 00:00:00
    Finished recover at 12-MAY-16

    database opened
    new incarnation of database registered in recovery catalog
    starting full resync of recovery catalog
    full resync complete
    repair failure complete
```

此时，Oracle 顺利完成数据库恢复，并以 resetlogs 方式重新打开数据库。

我们检查一下数据库的状态：

```
    SYS@ora11g> exit
    Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 -
64bit Production
    With the Partitioning, OLAP, Data Mining and Real Application Testing options
    [oracle@rhel6 ~]$ sqlplus / as sysdba
    SQL*Plus: Release 11.2.0.4.0 Production on Thu May 12 10:10:21 2016
    Copyright (c) 1982, 2013, Oracle.  All rights reserved.
    Connected to:
    Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
    With the Partitioning, OLAP, Data Mining and Real Application Testing options
    SYS@ora11g> select open_mode from v$database;
    OPEN_MODE
    --------------------
    READ WRITE
    SYS@ora11g> select status from v$instance;
    STATUS
    ------------
    OPEN
    SYS@ora11g> select group#,sequence#,status from v$log;

        GROUP#  SEQUENCE# STATUS
    ---------- --------- ----------------
             4          1 CURRENT
             5          0 UNUSED
             6          0 UNUSED
```

这样，整个数据库就恢复出来了。各位读者在完成本实验时，也可以自己先创建一些实验对象或者数据，然后进行数据库恢复，最后去检查一下先前创建的实验对象，看是否有数据丢

失现象。

5.14　本章涉及的相关概念

EM

所谓 EM，实际上就是 Oracle Enterprise Manager，是 Oracle 提供的一个用来对数据库进行监控和管理的图形化工具。该工具相当强大，尤其是对于初学者来说，使用 em 可以很容易地完成诸多数据库管理任务。我们这里对 em 的使用和配置做简单介绍。

ora11g 数据库是我们手工创建的，没有配置 em，catdb 则是我们使用 dbca 创建的，配置了 em，因此，这里使用 catdb 的 em。

首先启动 em:

```
[oracle@rhel6 ~]$ export ORACLE_SID=catdb
[oracle@rhel6 ~]$ emctl start dbconsole;
Oracle Enterprise Manager 11g Database Control Release 11.2.0.4.0
Copyright (c) 1996, 2013 Oracle Corporation.  All rights reserved.
https://rhel6:1158/em/console/aboutApplication
Starting Oracle Enterprise Manager 11g Database Control .......... started.
------------------------------------------------------------------
Logs are generated in directory
/u01/oracle/product/11.2.0/rhel6_catdb/sysman/log
```

启动后，会显示相应的 URL，就是上面的阴影着重显示部分。我们在该链接上点击右键，选择 Open Link，则会使用 firefox 打开链接，如图 5-19 所示。

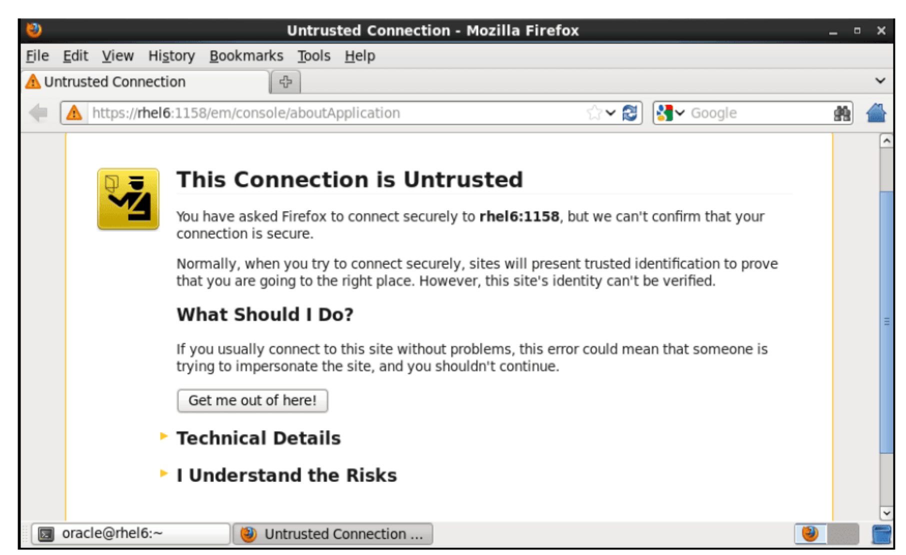


图 5-19　不信任链接页面

浏览器认为该链接是一个不受信任的非安全链接，因此弹出了这些警告信息。我们点击下面的 I Understand the Risks，则显示如下内容(图 5-20)：

I Understand the Risks

If you understand what's going on, you can tell Firefox to start trusting this site's identification. **Even if you trust the site, this error could mean that someone is tampering with your connection.**

Don't add an exception unless you know there's a good reason why this site doesn't use trusted identification.

Add Exception...

图 5-20　潜在风险页面

点击 Add Exception…，弹出如图 5-21 的窗口。



图 5-21　确认无风险页面

点击下面的 Confirm Security Exception，则进入到 em 的登录界面(图 5-22)：

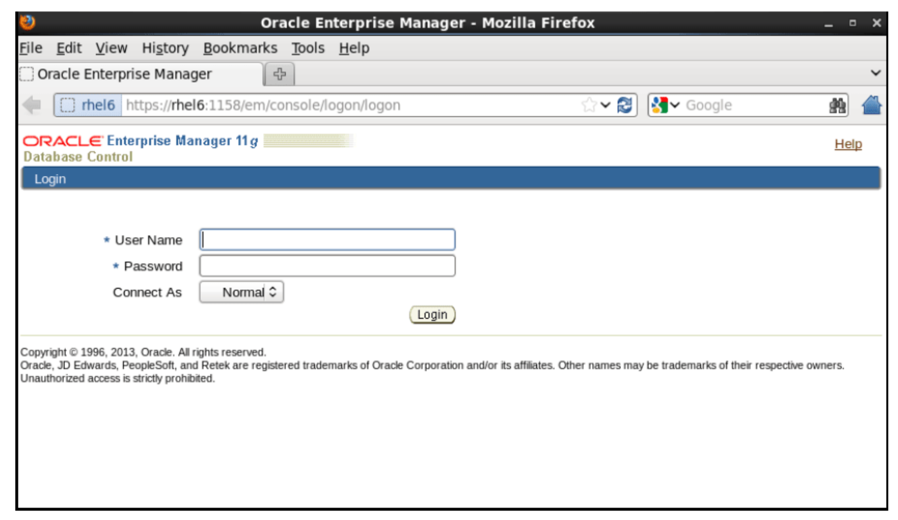


图 5-22　EM 登录页面

输入用户名 sys，密码 oracle，Connect As 选择 SYSDBA，点击 Login(图 5-23)：

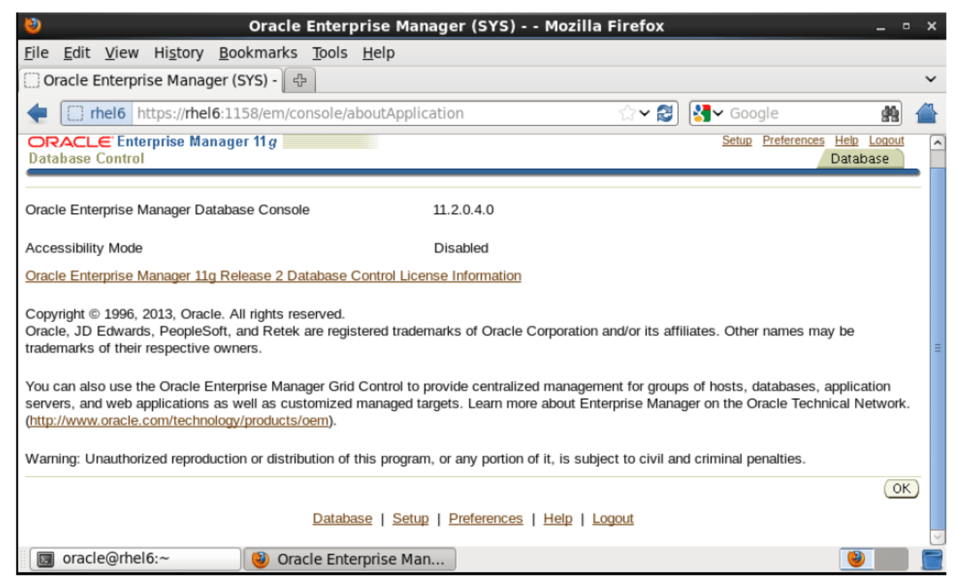


图 5-23　控制台页面

点击下面的 Database，则进入 em 的主界面(图 5-24)：

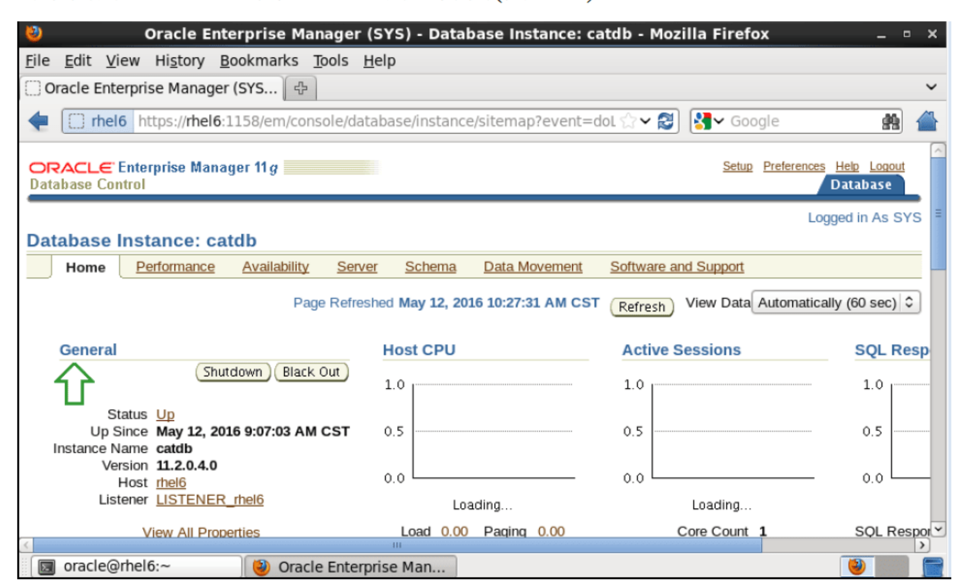


图 5-24　EM 主界面

在这里，可以查看当前数据库的状态，并进行数据库管理、维护以及优化工作。各位读者可以自行点击这里的标签，来查看相应的内容。

关于 em 更详细的用法，会在后续书籍中再做阐述。

em 的关闭：

```
[oracle@rhel6 ~]$ emctl stop dbconsole;
Oracle Enterprise Manager 11g Database Control Release 11.2.0.4.0
Copyright (c) 1996, 2013 Oracle Corporation. All rights reserved.
https://rhel6:1158/em/console/aboutApplication
Stopping Oracle Enterprise Manager 11g Database Control ...
 ... Stopped.
```

需要注意，em 通过监听连接到要管理的目标数据库，因此，在使用 em 之前，需要先启动目标数据库和监听。

隐含参数

在 Oracle 数据库中，除了可以使用 show parameter 命令查看的初始化参数之外，还有一类隐含参数。这些参数以下划线(_)开头，对 show parameter 命令不可见。但是，我们也可以用一种很简易方法来查看几个隐含参数：

```
SYS@ora11g> show parameter spfile;
NAME         TYPE         VALUE
----------- ----------- ------------------------------
spfile       string       /u01/oracle/product/11.2.0/dbs
                          /spfileora11g.ora
SYS@ora11g> create pfile from spfile;
File created.
```

然后，我们来查看生成的 pfile 文件的内容：

```
[oracle@rhel6 ~]$ cd $ORACLE_HOME/dbs
[oracle@rhel6 dbs]$ more initora11g.ora
ora11g.__db_cache_size=339738624
ora11g.__java_pool_size=4194304
ora11g.__large_pool_size=8388608
ora11g.__oracle_base='/u01/oracle'#ORACLE_BASE set
 from environment
ora11g.__pga_aggregate_target=352321536
ora11g.__sga_target=486539264
ora11g.__shared_io_pool_size=0
ora11g.__shared_pool_size=125829120
ora11g.__streams_pool_size=0
·······后面的内容
```

这些就是隐含参数。

关于隐含参数，需要注意，这些隐含参数有的是 Oracle 即将丢弃的，有的是 Oracle 为将来的数据库版本特性而预先准备的，有的用于测试或者其他特殊用途。因此，不建议在生产系

统中使用这些参数。除非你对它足够了解，或者是在 Oracle 的官方技术支持下使用。

scn

system change/commit number 称为 Oracle 数据库的系统改变号或者系统提交号。用来对数据库中的操作进行标示和排序，单向递增，类似于 Oracle 数据库中的时间戳。本书提供了两种方法来获得当前系统的 scn：

```
SYS@ora11g> select current_scn from v$database;
CURRENT_SCN
-----------
    432558
SYS@ora11g> select dbms_flashback.get_system_change_number from dual;
GET_SYSTEM_CHANGE_NUMBER
------------------------
   432562
```

5.15　本章用到的 Linux 命令

ps　用来显示当前系统中进程的信息
dd　用来创建或者备份文件
tar　压缩或者解压文件命令

l



第6章

性能优化系列实验

在实际工作中，对于大部分DBA而言，主要工作就是性能调整与优化。

毕竟，当系统上线后，运维才是最长期的工作。数据库不会不时出现数据丢失或者磁盘损坏的故障，更多的是数据库性能不佳，SQL执行速度太慢等问题。

在Oracle中，数据库是利用一个称为优化器的组件对用户提交的SQL进行分析，从而选择它认为的最优执行方式来执行SQL并返回结果。在11*g*版本中，默认的优化器为CBO(Cost-Based Optimizer)，称为基于成本的优化器。一条SQL可能会有多种访问数据和执行的方式，

但 Oracle 通过成本(cost)来衡量哪种方式最佳。所谓成本，指的就是不同的执行方式消耗的物理 I/O、内存以及 CPU 资源的数量。

本书旨在为初学者提供能够快速上手的相关实验，使得读者能够尽快掌握 Oracle DBA 的一些基本动手操作能力。所以本章并不深入探讨 CBO 以及优化器，而只是先描述一些相关的初步知识，更深入的内容敬请关注后续书籍。

6.1 统计信息收集实验

为生成最优的执行计划，优化器需要事先知道一些相关信息。比如说，如果想通过把一张表的数据从头到尾全部扫描一遍来得到最终结果，那么优化器就需要知道这个表有多少条记录，平均每条记录有多长，一共占用了多少个 block。这样，优化器才能够评估扫描这个表需要进行几次物理 I/O，也就是可以估算出相应的成本。那么，优化器如何知道这些信息呢？

默认情况下，Oracle 会自动收集数据库中对象的相关信息。这些信息称为统计信息。但这种自动收集机制往往无法满足实际要求，因为这些自动任务都是在每晚固定时刻运行。对于那些在一天内数据变化特别频繁的对象而言，这样的收集统计信息的机制显然是不够的。我们需要手工收集。

先看表的统计信息收集。

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> create table tab_t1 as select * from user_objects;
Table created.
```

我们先创建一个测试表 tab_t1，然后查看数据库中关于该表的统计信息。实际上，数据库中各个对象的统计信息就记录在对应的数据字典表中：

```
SCOTT@ora11g> desc user_tables;
 Name                      Null?      Type
 ------------------------- ---------- --------------------------
 TABLE_NAME                NOT NULL   VARCHAR2(30)
 TABLESPACE_NAME                      VARCHAR2(30)
 CLUSTER_NAME                         VARCHAR2(30)
 IOT_NAME                             VARCHAR2(30)
 STATUS                               VARCHAR2(8)
 PCT_FREE                             NUMBER
 PCT_USED                             NUMBER
 INI_TRANS                            NUMBER
 MAX_TRANS                            NUMBER
 INITIAL_EXTENT                       NUMBER
```

```
 NEXT_EXTENT                                   NUMBER
 MIN_EXTENTS                                   NUMBER
 MAX_EXTENTS                                   NUMBER
 PCT_INCREASE                                  NUMBER
 FREELISTS                                     NUMBER
 FREELIST_GROUPS                               NUMBER
 LOGGING                                       VARCHAR2(3)
 BACKED_UP                                     VARCHAR2(1)
NUM_ROWS                                       NUMBER
BLOCKS                                         NUMBER
EMPTY_BLOCKS                                   NUMBER
AVG_SPACE                                      NUMBER
CHAIN_CNT                                      NUMBER
AVG_ROW_LEN                                    NUMBER
 AVG_SPACE_FREELIST_BLOCKS                     NUMBER
 NUM_FREELIST_BLOCKS                           NUMBER
 DEGREE                                        VARCHAR2(40)
 INSTANCES                                     VARCHAR2(40)
 CACHE                                         VARCHAR2(20)
 TABLE_LOCK                                    VARCHAR2(8)
 SAMPLE_SIZE                                   NUMBER
 LAST_ANALYZED                                 DATE
 PARTITIONED                                   VARCHAR2(3)
 IOT_TYPE                                      VARCHAR2(12)
 TEMPORARY                                     VARCHAR2(1)
 SECONDARY                                     VARCHAR2(1)
 NESTED                                        VARCHAR2(3)
 BUFFER_POOL                                   VARCHAR2(7)
 FLASH_CACHE                                   VARCHAR2(7)
 CELL_FLASH_CACHE                              VARCHAR2(7)
 ROW_MOVEMENT                                  VARCHAR2(8)
 GLOBAL_STATS                                  VARCHAR2(3)
 USER_STATS                                    VARCHAR2(3)
 DURATION                                      VARCHAR2(15)
 SKIP_CORRUPT                                  VARCHAR2(8)
 MONITORING                                    VARCHAR2(3)
 CLUSTER_OWNER                                 VARCHAR2(30)
 DEPENDENCIES                                  VARCHAR2(8)
 COMPRESSION                                   VARCHAR2(8)
 COMPRESS_FOR                                  VARCHAR2(12)
```

```
 DROPPED                              VARCHAR2(3)
 READ_ONLY                            VARCHAR2(3)
 SEGMENT_CREATED                      VARCHAR2(3)
 RESULT_CACHE                         VARCHAR2(7)
```

阴影着重显示部分就是与表相关的统计信息。我们看一下关于表 tab_t1 的统计信息：

```
SCOTT@ora11g> select NUM_ROWS,BLOCKS,EMPTY_BLOCKS,AVG_SPACE,CHAIN_CNT,
AVG_ROW_LEN from user_tables where table_name='TAB_T1';
  NUM_ROWS     BLOCKS EMPTY_BLOCKS  AVG_SPACE  CHAIN_CNT AVG_ROW_LEN
---------- ---------- ------------ ---------- ---------- -----------
```

此时结果为空，也就是说，在创建表的时候，Oracle 不会自动收集该表的统计信息。我们需要手工收集：

```
SCOTT@ora11g> exec dbms_stats.gather_table_stats('SCOTT','TAB_T1');
PL/SQL procedure successfully completed.
```

这里使用 Oracle 提供的程序来收集。关于 DBMS_STATS 的详细内容，可参考官方文档 Database PL/SQL Packages and Types Reference 中的第 142 章。

然后查看一下统计信息：

```
SCOTT@ora11g> select NUM_ROWS,BLOCKS,EMPTY_BLOCKS,AVG_SPACE,CHAIN_CNT,
AVG_ROW_LEN from user_tables where table_name='TAB_T1';
  NUM_ROWS     BLOCKS EMPTY_BLOCKS  AVG_SPACE  CHAIN_CNT AVG_ROW_LEN
---------- ---------- ------------ ---------- ---------- -----------
        16          1            0          0          0          82
```

其中，NUM_ROWS 表示表中共有多少条记录；BLOCKS 表示表使用了多少个数据块来存储数据；EMPTY_BLOCKS 表示表中有多少个空 block；AVG_SPACE 表示存储数据的 block 中的平均剩余空间；CHAIN_CNT 表示发生行迁移或者行链接的记录数；AVG_ROW_LEN 表示记录的平均长度。

关于行迁移或者行链接的内容，我们将在后续书籍中讲解。

有了这些统计信息，Oracle 在估算 SQL 查询的成本时，就有了信息基础。需要注意，准确的统计信息是优化器生成最优执行方式的关键条件。

再来看索引。

```
SCOTT@ora11g> create index ind_t1_id on tab_t1(object_id);
Index created.
```

索引的统计信息也存储在数据字典表中：

```
SCOTT@ora11g> desc user_indexes;
```

```
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INDEX_NAME                                NOT NULL VARCHAR2(30)
 INDEX_TYPE                                         VARCHAR2(27)
 TABLE_OWNER                               NOT NULL VARCHAR2(30)
 TABLE_NAME                                NOT NULL VARCHAR2(30)
 TABLE_TYPE                                         VARCHAR2(11)
 UNIQUENESS                                         VARCHAR2(9)
 COMPRESSION                                        VARCHAR2(8)
 PREFIX_LENGTH                                      NUMBER
 TABLESPACE_NAME                                    VARCHAR2(30)
 INI_TRANS                                          NUMBER
 MAX_TRANS                                          NUMBER
 INITIAL_EXTENT                                     NUMBER
 NEXT_EXTENT                                        NUMBER
 MIN_EXTENTS                                        NUMBER
 MAX_EXTENTS                                        NUMBER
 PCT_INCREASE                                       NUMBER
 PCT_THRESHOLD                                      NUMBER
 INCLUDE_COLUMN                                     NUMBER
 FREELISTS                                          NUMBER
 FREELIST_GROUPS                                    NUMBER
 PCT_FREE                                           NUMBER
 LOGGING                                            VARCHAR2(3)
BLEVEL                                              NUMBER
LEAF_BLOCKS                                         NUMBER
DISTINCT_KEYS                                       NUMBER
AVG_LEAF_BLOCKS_PER_KEY                             NUMBER
AVG_DATA_BLOCKS_PER_KEY                             NUMBER
CLUSTERING_FACTOR                                   NUMBER
 STATUS                                             VARCHAR2(8)
 NUM_ROWS                                           NUMBER
 SAMPLE_SIZE                                        NUMBER
 LAST_ANALYZED                                      DATE
 DEGREE                                             VARCHAR2(40)
 INSTANCES                                          VARCHAR2(40)
 PARTITIONED                                        VARCHAR2(3)
 TEMPORARY                                          VARCHAR2(1)
 GENERATED                                          VARCHAR2(1)
 SECONDARY                                          VARCHAR2(1)
```

```
BUFFER_POOL                        VARCHAR2(7)
FLASH_CACHE                        VARCHAR2(7)
CELL_FLASH_CACHE                   VARCHAR2(7)
USER_STATS                         VARCHAR2(3)
DURATION                           VARCHAR2(15)
PCT_DIRECT_ACCESS                  NUMBER
ITYP_OWNER                         VARCHAR2(30)
ITYP_NAME                          VARCHAR2(30)
PARAMETERS                         VARCHAR2(1000)
GLOBAL_STATS                       VARCHAR2(3)
DOMIDX_STATUS                      VARCHAR2(12)
DOMIDX_OPSTATUS                    VARCHAR2(6)
FUNCIDX_STATUS                     VARCHAR2(8)
JOIN_INDEX                         VARCHAR2(3)
IOT_REDUNDANT_PKEY_ELIM            VARCHAR2(3)
DROPPED                            VARCHAR2(3)
VISIBILITY                         VARCHAR2(9)
DOMIDX_MANAGEMENT                  VARCHAR2(14)
SEGMENT_CREATED                    VARCHAR2(3)
```

上述阴影着重显示部分就是关于索引的统计信息。我们来看一下索引 ind_t1_id 的统计信息：

```
    SCOTT@ora11g> select BLEVEL,LEAF_BLOCKS,DISTINCT_KEYS,
AVG_LEAF_BLOCKS_PER_KEY,AVG_DATA_BLOCKS_PER_KEY,CLUSTERING_FACTOR from
user_indexes where index_name='IND_T1_ID';
       BLEVEL LEAF_BLOCKS DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY
    ---------- ----------- ------------- -----------------------
    AVG_DATA_BLOCKS_PER_KEY CLUSTERING_FACTOR
    ----------------------- -----------------
      0     1    16      1
            1    1
```

这里是有记录的，也就是说，在创建索引的时候，Oracle 会自动收集其统计信息。其中，BLEVEL 表示索引的深度(因为默认索引类型的结构是一个树形结构)；LEAF_BLOCKS 表示索引中叶子块的数目(在索引中，叶子块才真正存储索引数据)；DISTINCT_KEYS 表示索引列中的不同值的个数；AVG_LEAF_BLOCKS_PER_KEY 表示叶子块上平均存储的索引值的个数，这里的索引值也称为键值；AVG_DATA_BLOCKS_PER_KEY 表示每个数据库上存储的键值的个数；CLUSTERING_FACTOR 则反映了索引中数据和表中数据的顺序对应关系。

当然，也可以手工收集索引的统计信息。

```
SCOTT@ora11g> exec dbms_stats.gather_index_stats('SCOTT','IND_T1_ID');
PL/SQL procedure successfully completed.
```

当然还可以收集整个用户下所有对象的统计信息：

```
SCOTT@ora11g> exec dbms_stats.gather_schema_stats('SCOTT');
PL/SQL procedure successfully completed.
```

甚至收集整个数据库的统计信息：

```
SCOTT@ora11g> exec dbms_stats.gather_database_stats;
BEGIN dbms_stats.gather_database_stats; END;
*
ERROR at line 1:
ORA-20000: Insufficient privileges to analyze an object in Database
ORA-06512: at "SYS.DBMS_STATS", line 25335
ORA-06512: at "SYS.DBMS_STATS", line 25511
ORA-06512: at "SYS.DBMS_STATS", line 25467
ORA-06512: at line 1
```

权限不够，我们需要使用 sys 用户：

```
SCOTT@ora11g> conn / as sysdba
Connected.
SYS@ora11g> exec dbms_stats.gather_database_stats;
PL/SQL procedure successfully completed.
```

当然，收集全库统计信息这样的动作，在生产库中尤其要注意，因为全库中对象较多，这样的操作会消耗很长时间，甚至会影响数据库中现有 SQL 的执行。因此，如果确有必要收集全库的统计信息，建议放在数据库较空闲的时候来做，如凌晨。

6.2　索引访问方式实验

前面提到了基本的索引创建实验，通过索引，我们在访问某些大表中的一小部分数据时，能加快执行的速度。那么，通过索引访问数据，又有哪些方式呢？

我们来看实验：

```
SYS@ora11g> conn scott/tiger;
Connected.
SCOTT@ora11g> set autot traceonly;
SP2-0618: Cannot find the Session Identifier.  Check PLUSTRACE role is enabled
SP2-0611: Error enabling STATISTICS report
```

这里的 set autot traceonly 命令指我们想在 scott 用户下打开自动跟踪，从而查看 SQL 的执行计划。所谓的执行计划，其实就是 Oracle 在处理用户提交的 SQL 时，真正访问数据的方式。

但这里报错了，我们需要进行如下处理：

```
SCOTT@ora11g> @?/rdbms/admin/utlxplan.sql;
Table created.
```

然后切换到 sys 用户：

```
SCOTT@ora11g> conn / as sysdba
Connected.
```

执行脚本创建 plustrace 角色：

```
SYS@ora11g> @?/sqlplus/admin/plustrce.sql;
SYS@ora11g>
SYS@ora11g> drop role plustrace;
drop role plustrace
        *
ERROR at line 1:
ORA-01919: role 'PLUSTRACE' does not exist
SYS@ora11g> create role plustrace;
Role created.
SYS@ora11g>
SYS@ora11g> grant select on v_$sesstat to plustrace;
Grant succeeded.
SYS@ora11g> grant select on v_$statname to plustrace;
Grant succeeded.
SYS@ora11g> grant select on v_$mystat to plustrace;
Grant succeeded.
SYS@ora11g> grant plustrace to dba with admin option;
Grant succeeded.
SYS@ora11g>
SYS@ora11g> set echo off
```

将该角色授予 scott 用户：

```
SYS@ora11g> grant plustrace to scott;
Grant succeeded.
```

这样就可以使用了：

```
SYS@ora11g> conn scott/tiger;
```

```
Connected.
```

接着创建测试表 tab_exp：

```
SCOTT@ora11g> create table tab_exp ( id number,name varchar2(30));
Table created.
```

然后插入数据：

```
SCOTT@ora11g>
begin
  for i in 1..10000
  loop
  insert into tab_exp values (i,'test'||i);
  end loop;
  commit;
end;
/
PL/SQL procedure successfully completed.
```

并在该表上创建索引：

```
SCOTT@ora11g> create index ind_exp_id on tab_exp(id);
Index created.
```

然后执行如下 SQL 并查看其执行计划：

```
SCOTT@ora11g> set autot traceonly;
SCOTT@ora11g> select * from tab_exp where id =100;
Execution Plan
----------------------------------------------------------
Plan hash value: 3983777909
----------------------------------------------------------------- ----------
| Id  | Operation      | Name  | Rows  | Bytes | Cost (%CPU)| Time |
----------------------------------------------------------------- ----------
|   0 | SELECT STATEMENT    |    |     1 |    30 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| TAB_EXP |     1 |     30 |      2   (0)|
00:00:01 |
|*  2 |   INDEX RANGE SCAN     | IND_EXP_ID |     1 | |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("ID"=100)
```

```
Note
-----
   - dynamic sampling used for this statement (level=2)
Statistics
----------------------------------------------------------
  10  recursive calls
   0  db block gets
  41  consistent gets
   1  physical reads
   0  redo size
 598  bytes sent via SQL*Net to client
 523  bytes received via SQL*Net from client
   2  SQL*Net roundtrips to/from client
   0  sorts (memory)
   0  sorts (disk)
   1  rows processed
```

上述命令的输出内容较多，我们暂不做全部解释，只看阴影着重显示部分，表示我们使用索引来访问数据，并且使用了索引范围扫描(index range scan)。索引中的数据默认按由小到大的升序模式进行存储，因此在执行这里提交的 SQL 时，Oracle 会在索引中先找到 id 为 100 的记录，然后继续向下扫描，一直找到下一个不为 100 的记录为止，因此这里是范围扫描。

但这里要注意，实际上表 tab_exp 中 id=100 的记录只有一条，也就是说如果 Oracle 知道这一点的话，它就根本不需要去进行范围扫描了。我们删除原有的索引，然后创建唯一性索引：

```
SCOTT@ora11g> set autot off;
SCOTT@ora11g> drop index ind_exp_id;
Index dropped.
SCOTT@ora11g> create unique index ind_exp_id on tab_exp(id);
Index created.
```

然后我们再来查看同样 SQL 的执行计划：

```
SCOTT@ora11g> set autot traceonly;
SCOTT@ora11g> select * from tab_exp where id = 100;
Execution Plan
----------------------------------------------------------
Plan hash value: 960575795
--------------------------------------------------------------------------
| Id  | Operation        | Name | Rows  | Bytes | Cost (%CPU)|Time |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |    |     1 |    30 |     2   (0)|00:00:01 |
```

```
|   1 |  TABLE ACCESS BY INDEX ROWID| TAB_EXP |     1 |    30 |     2
(0)|00:00:01 |
|*  2 |   INDEX UNIQUE SCAN    | IND_EXP_ID |     1 | |     1   (0)|00:00:01 |
--------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("ID"=100)
Statistics
----------------------------------------------------------
   3  recursive calls
   0  db block gets
   6  consistent gets
   1  physical reads
   0  redo size
 462  bytes sent via SQL*Net to client
 512  bytes received via SQL*Net from client
   1  SQL*Net roundtrips to/from client
   0  sorts (memory)
   0  sorts (disk)
   1  rows processed
```

此时，索引的访问方式就变成唯一性扫描(index unique scan)。

6.3　数据访问方式实验

前面我们提到，当 Oracle 执行 SQL 去访问数据时，可使用全表扫描(table access full)，也可以使用索引(例如唯一性扫描或者范围扫描，当然还有其他索引扫描方式)。而实际上，除这两种方式外，还有一种方式：使用 rowid。

rowid 是 Oracle 提供的用来直接访问数据的方式，rowid 中记录了数据所在的文件编号、所在的 block 编号以及行号。这样，访问数据的速度就会更快：

```
SCOTT@ora11g> set autot off;
SCOTT@ora11g> select id,name,rowid from tab_exp where id = 99;
 ID NAME      ROWID
---------- ------------------------------ ------------------
 99 test99     AAADalAAEAAAAEBABi
```

我们先找出 id=99 的这条记录的 rowid。rowid 并不存在于表中，它只是在你执行 sql 的时候，由 oracle 自动生成。

```
SCOTT@ora11g> set autot traceonly;
SCOTT@ora11g> select * from tab_exp where rowid = 'AAADalAAEAAAAEBABi';
Execution Plan
----------------------------------------------------------
Plan hash value: 1044868054
--------------------------------------------------------------------------
| Id  | Operation      | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |    1 |   42 |    1   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY USER ROWID| TAB_EXP |    1 |   42 |    1   (0)|
00:00:01 |
--------------------------------------------------------------------------
Statistics
----------------------------------------------------------
   1  recursive calls
   0  db block gets
   1  consistent gets
   0  physical reads
   0  redo size
 593  bytes sent via SQL*Net to client
 523  bytes received via SQL*Net from client
   2  SQL*Net roundtrips to/from client
   0  sorts (memory)
   0  sorts (disk)
   1  rows processed
SCOTT@ora11g> set autot off;
```

需要注意，rowid 是数据存储的物理位置，因此一旦移动了表，记录的 rowid 也会随之改变。